
## Welcome To

# ISSCC 2014
# Session 27
# Energy-Efficient Digital Circuits

# A 460MHz@397mV – 2.6GHz@1.3V 32bit VLIW DSP Embedding $F_{MAX}$ Tracking

R. Wilson[1], E. Beigne[2], P. Flatresse[1], A. Valentian[2], F. Abouzeid[1], T. Benoist[2], C. Bernard[2], S. Bernard[2], O. Billoint[2], S. Clerc[1], B. Giraud[2], A. Grover[1], J. Le Coz[1], I. Miro-Panades[2], J.P. Noel[1], B. Pelloux-Prayer[1], P. Roche[1], O. Thomas[2], Y. Thonnart[2], D. Turgis[1], F. Clermidy[2], P. Magarshack[1]

*[1]STMicroelectronics, Crolles, France*

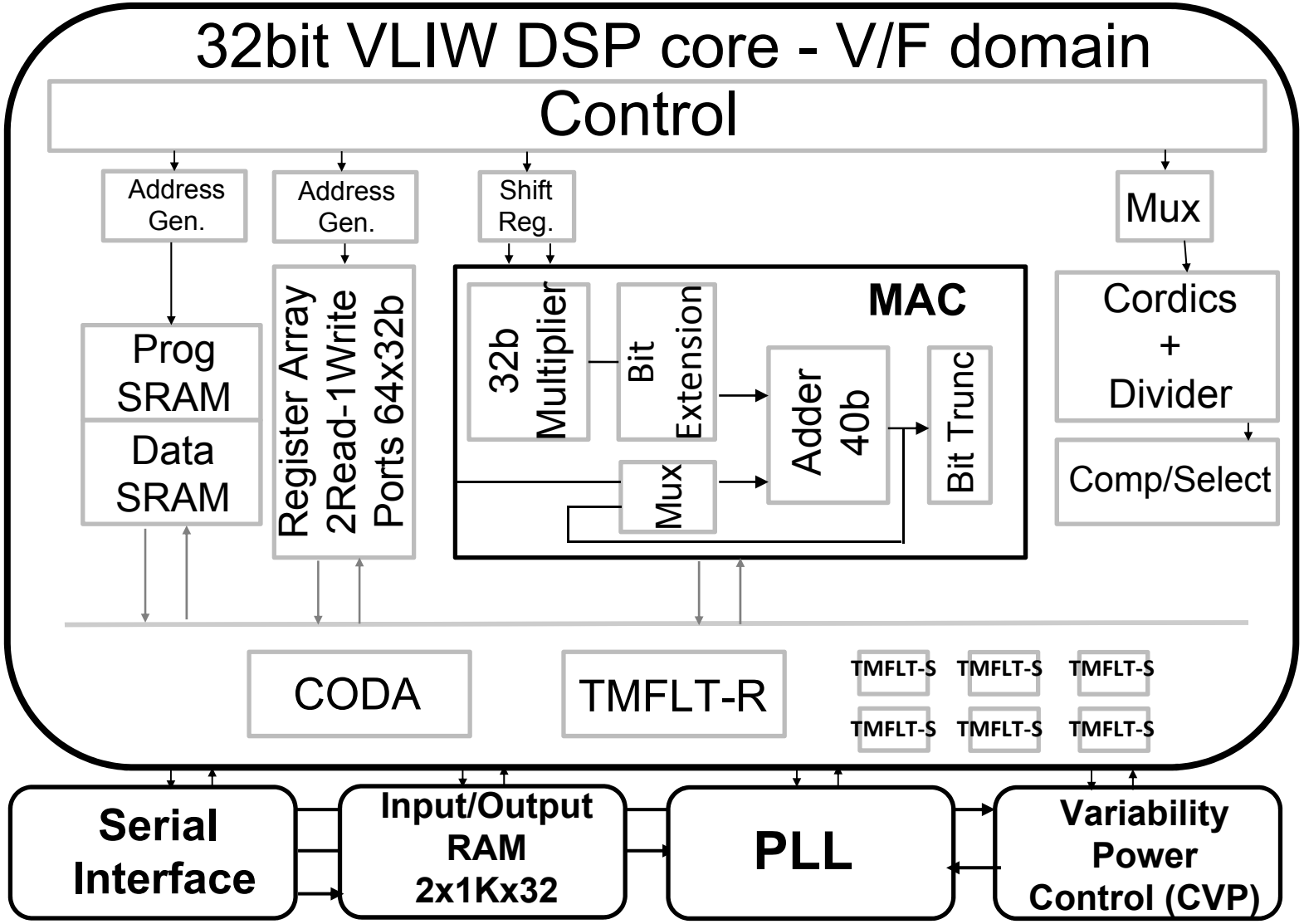
*[2]CEA-LETI, MINATEC, Grenoble, France*

# Application Context
## Wide Operating Range

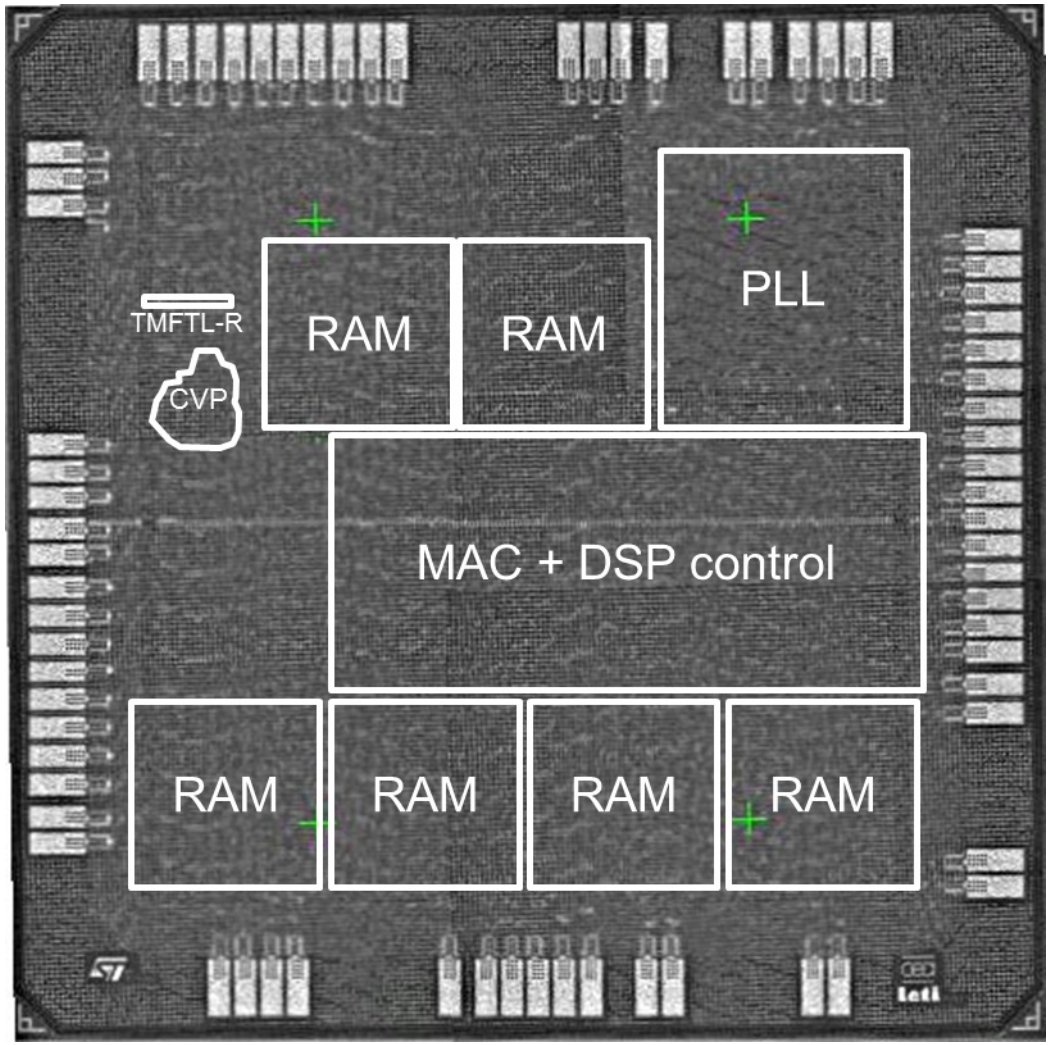
- Today's wireless internet applications:
  - Multimedia, gaming, GPS…
  - High speed > 1.5GHz
  - Large range voltage operation
  - Low standby power

- Challenges :
  - Ultra Wide Voltage Range (UWVR) : Versatility
  - Very high speed at nominal and low supply voltages
  - Low energy

# DSP architecture



# Chip Micrograph




| | |
|---|---|
| Technology STMicro | UTBB FDSOI 28 nm |
| Transistors | Flip-Well (LVT) L=24nm |
| Core area | 1 mm² |
| DSP benchmark | FFT 1024 |
| VDD range | 0.397V-1.3V |
| VBB range | 0V/±2V |

# Outline

## Achieving Performance in **UWVR**

- UTBB FDSOI 28 nm technology
  - Well-type methodology
  - Body Biasing techniques
- Optimized library sub-set
  - Assymetric standard cells
  - High performance flip-flops
- $F_{MAX}$ tracking techniques
  - Replica path and Timing fault detection
- DSP performances silicon results

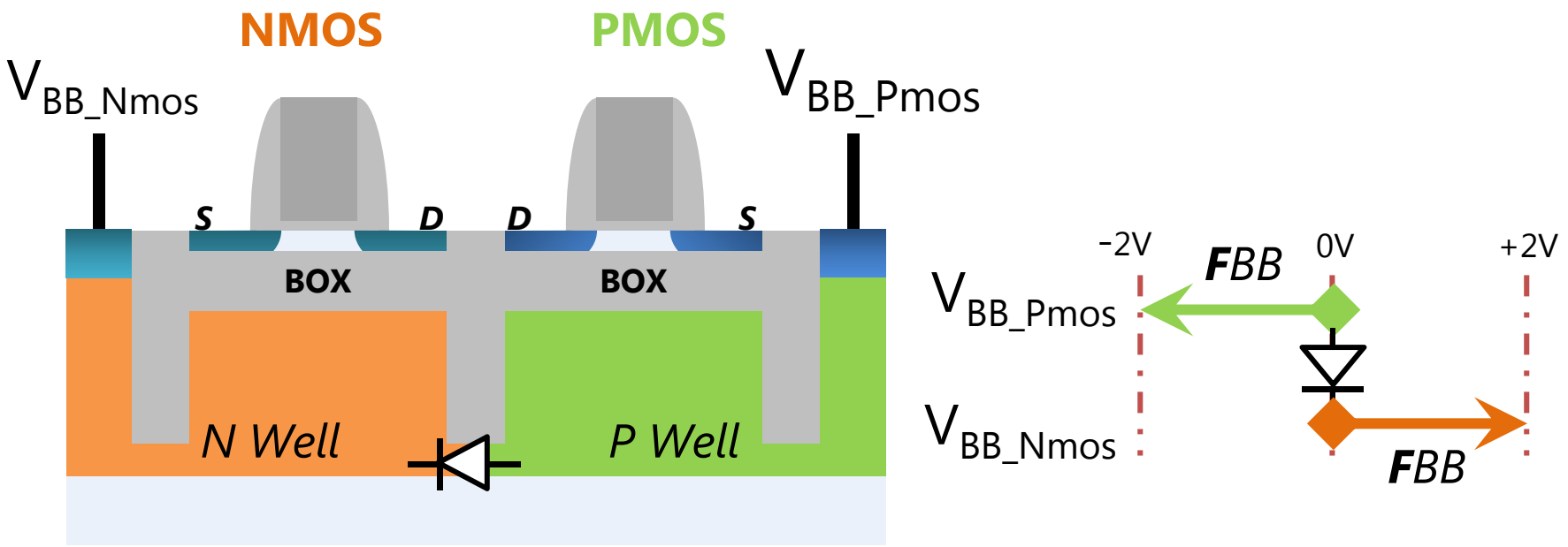
# Outline

## Achieving Performance in **UWVR**

- UTBB FDSOI 28 nm technology
  - Well-type methodology
  - Body Biasing techniques
- Optimized library sub-set
  - Assymetric standard cells
  - High performance flip-flops
- $F_{MAX}$ tracking techniques
  - Replica path and Timing fault detection
- DSP performance silicon results

# VT flavor choice - Well type

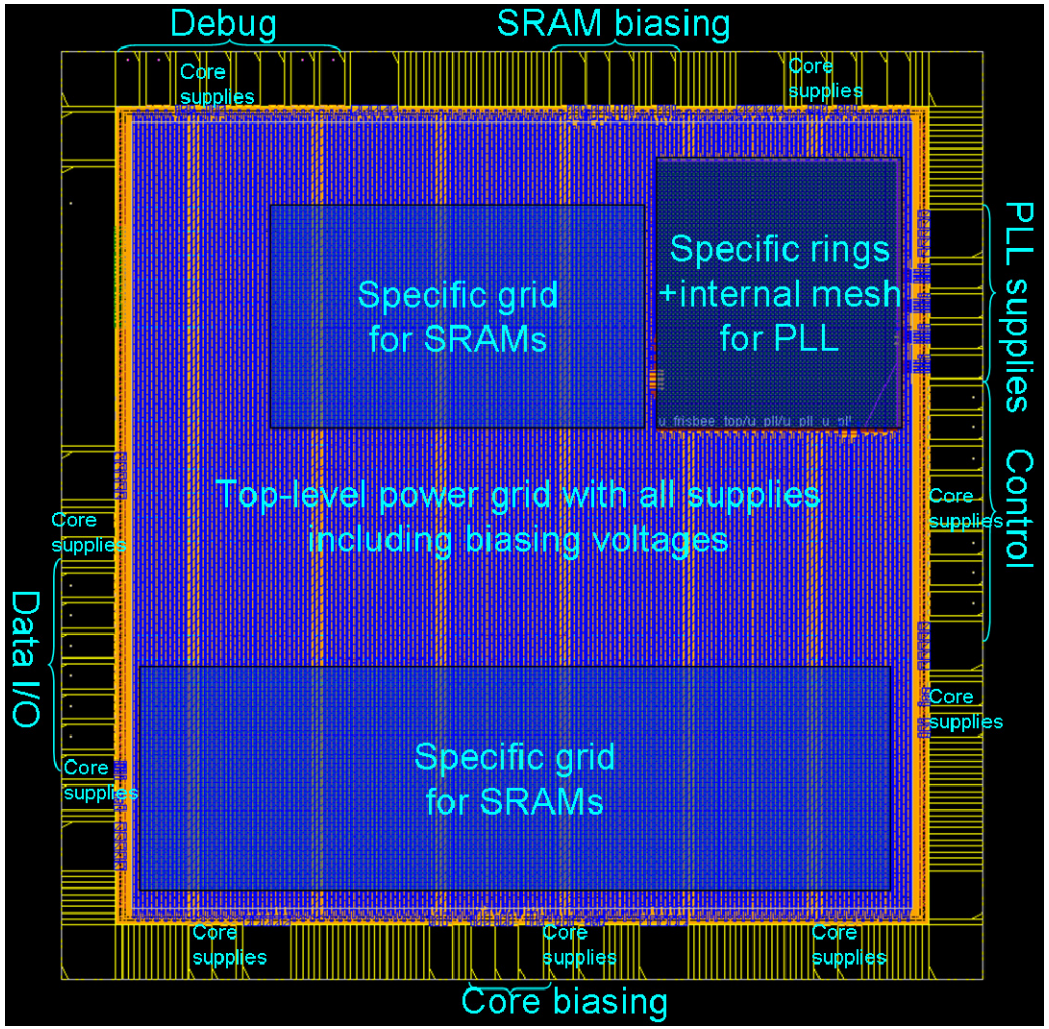
- UTBB FDSOI Low-VT transistors chosen for maximum speed at low voltage : Flip Well FDSOI




⇨ DSP core $V_{DDMIN}$ ↘ 397mV

⇨ Clock frequency ↗ +400%@500mV and +100%@1.3V

# Power distribution – Body Biasing




- Including all biasing voltages
  - Thin VBB power stripes
- Body-Biasing
  - Supplied through specific IOs (± 2V)

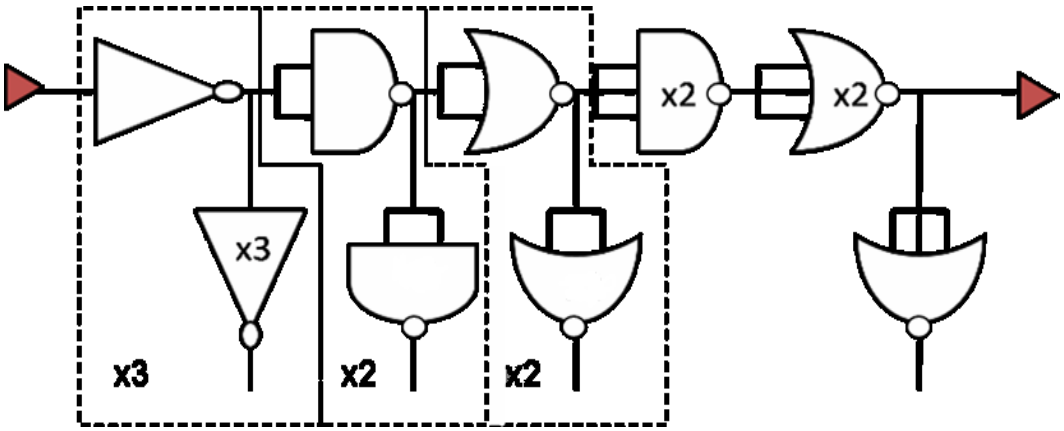
# Outline

## Achieving Performance in **UWVR**

- UTBB FDSOI 28 nm technology
  - Well-type methodology
  - Body Biasing techniques
- Optimized library sub-set
  - Assymetric standard cells
  - High performance flip-flops
- $F_{MAX}$ tracking techniques
  - Replica path and Timing fault detection
- DSP performances silicon results

# Optimized library sub-set

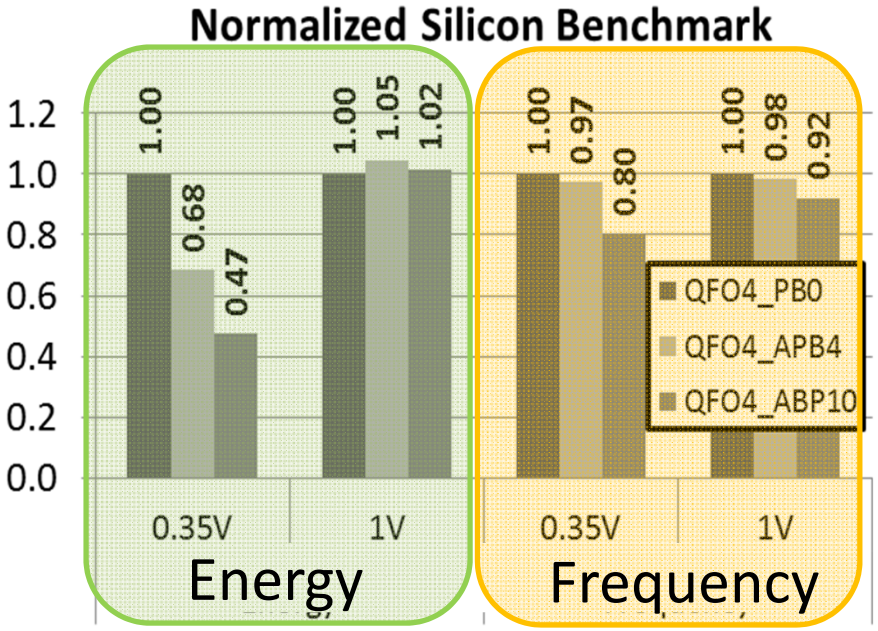
- Use of available standard cells optimized for low voltage:
  - Extend the subset to the Wide Voltage Range
- Gate length modulation done through **Poly Biasing** (PB)
  - Symetric/Assymetric Poly Biasing
- SPICE simulations and silicon measurements of qFO4 ring oscillators




qFO4

# Assymetric standard cells

- Keep the power-delay optimal cells in UWVR
  - For Low-VT cells, APB4 offers 30% energy gain with less than 3% frequency cost
- Cells are characterized over [0.275V-1.4V] $V_{DD}$ range and [0V- (±2V)] $V_{BB}$ range

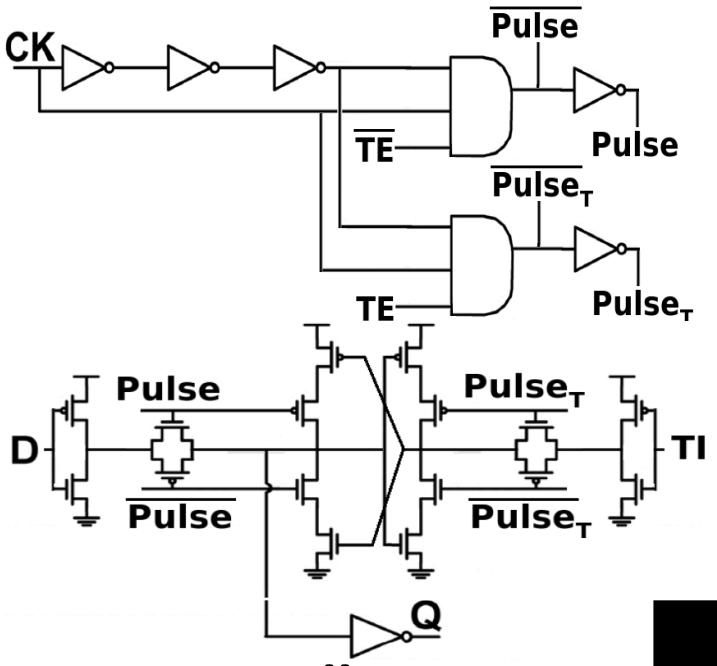



*FMAX Tracking*

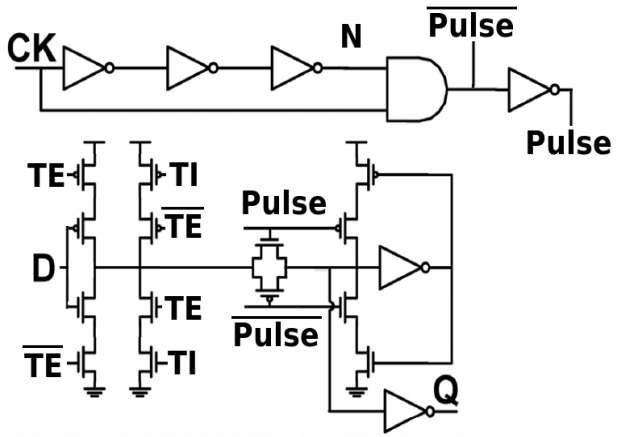
# Optimized Pulsed-latch FF

- Transmission-Gate Pulsed Latches
  - A latch and a pulse generator
  - Same behavior as a conventional Master-Slave FF
  - Proved to be the most energy-efficient in a large portion of the design space
- Two types of Transmission-Gate Pulsed Latches (TGPL) were implemented and compared to:
  - Classical C2MOS Flip-Flop
  - High-speed SA-based Flip-Flop

# Optimized Pulsed-latch FF




**TGPLmuxClk**




**TGPLmuxScan**

| | D-to-Q (ps) | Egy/cycle (fJ) | Area (µm²) |
|---|---|---|---|
| ST C2MOS | 117.5 | 6 | 4,41 |
| ST SA | 46.5 (- 60 %) | 12.5 (+ 208 %) | 6,85 (+ 55 %) |
| TGPLMuxScan | 30.5 **(- 74 %)** | 7.2 **(+ 20 %)** | 4,73 **(+ 7 %)** |
| TGPLMuxClk | 26 (- 78 %) | 9.1 (+ 56 %) | 5,06 (+ 14 %) |

# Outline

## Achieving Performance in **UWVR**

- UTBB FDSOI 28 nm technology
  - Well-type methodology
  - Body Biasing techniques
- Optimized library sub-set
  - Assymetric standard cells
  - High performance flip-flops
- $F_{MAX}$ tracking techniques
  - Replica path and Timing fault detection
- DSP performances silicon results

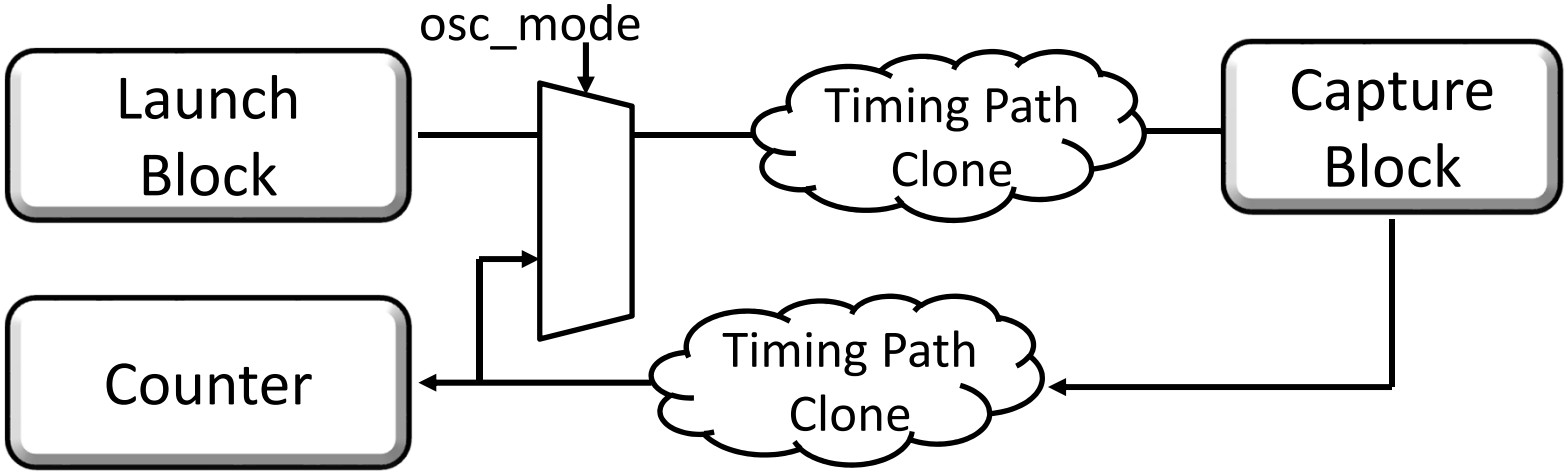
# Timing margins reduction in UWVR $F_{MAX}$ tracking

- Design in *typical* corner case instead of worst case approach
  - Need to compensate for PVT variations in UWVR
  - Reduce energy per operation … or increase clock frequency
- Track the optimal Frequency/Voltage/Biasing point
  - Real maximum frequency is **not** available
  - Functional in the UWVR / Low area and power overhead

- Two complementary solutions proposed on-chip
  - Replica path based ⇨**CODA (C**l**O**ning **DA**ta paths)
  - Timing slack sensor based ⇨ **TMFLT (T**i**M**ing **F**au**LT)**

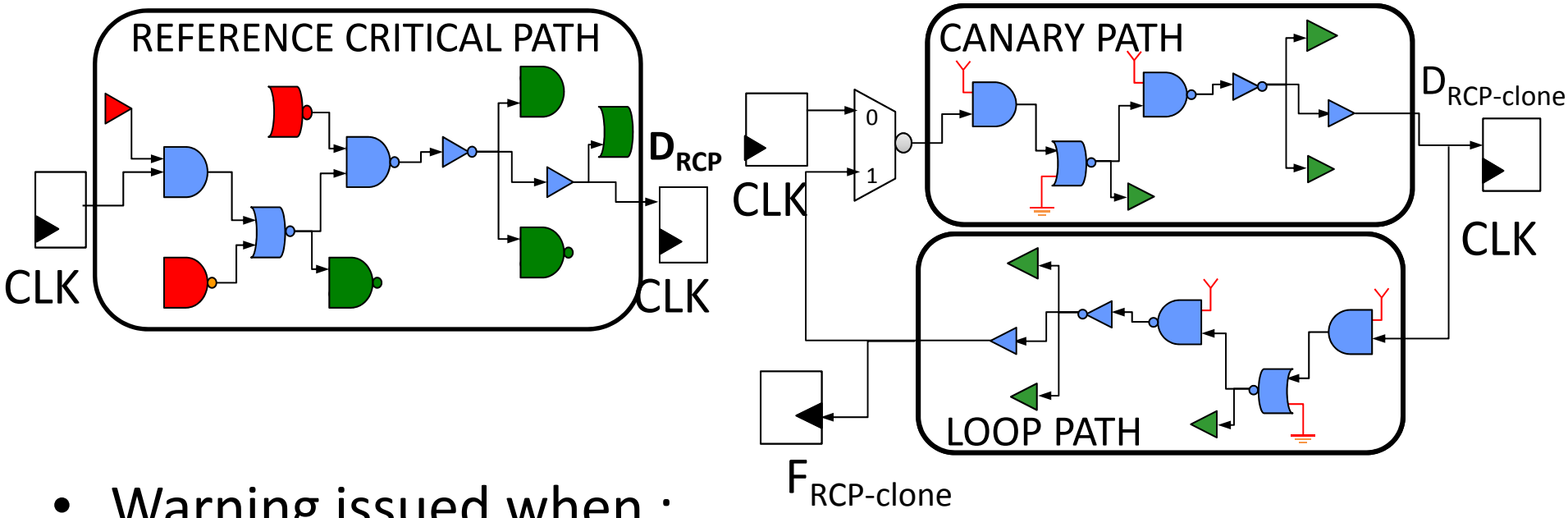
# $F_{MAX}$ tracking : CODA

- 16 pairs of clones of Representative Critical Paths (RCP)
  - One RCP is used as a canary path (delay monitor)
  - Another RCP used as a loop (oscillation frequency monitor)



# $F_{MAX}$ tracking : CODA

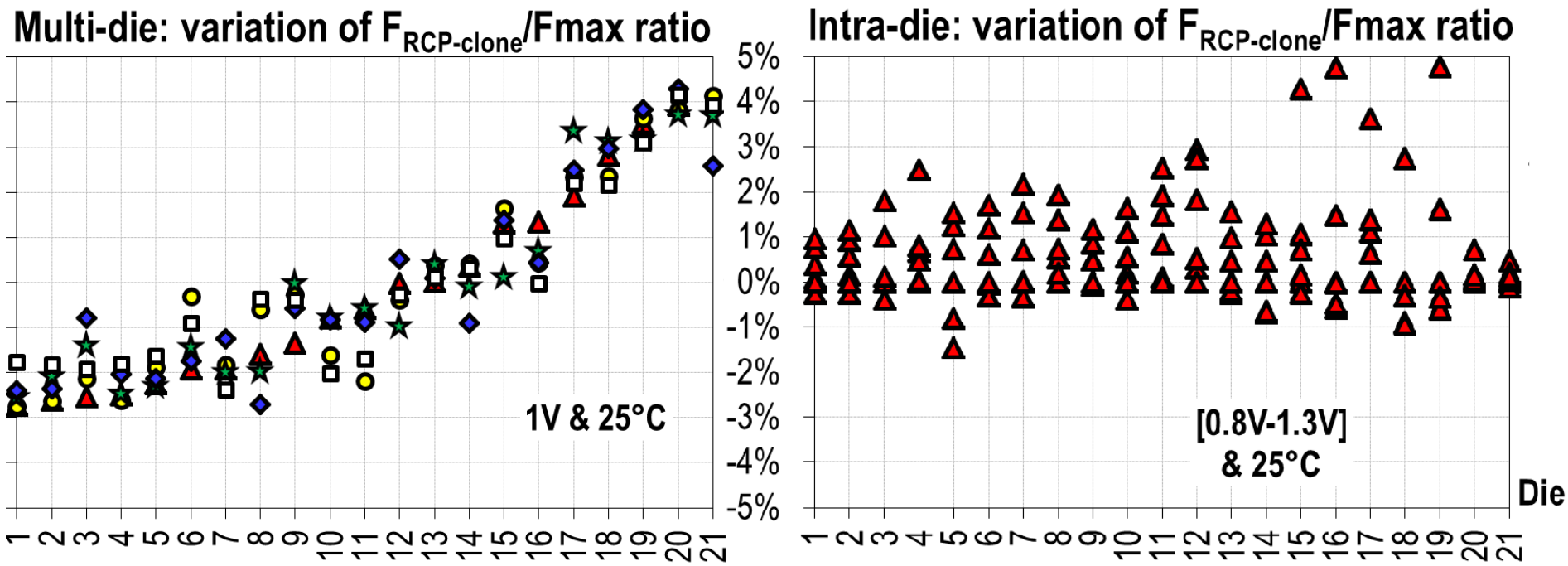
- 16 pairs of clones of Representative Critical Paths (RCP)
  - One RCP is used as a canary path (delay monitor)
  - Another RCP used as a loop (oscillation frequency monitor)




- Warning issued when :
  - $F_{CLK}=1/DRCP_{-clone}$ (< 1/DRCP)
  - correlated in loop mode through direct measurement of $F_{RCP-clone}$

# $F_{MAX}$ tracking : CODA

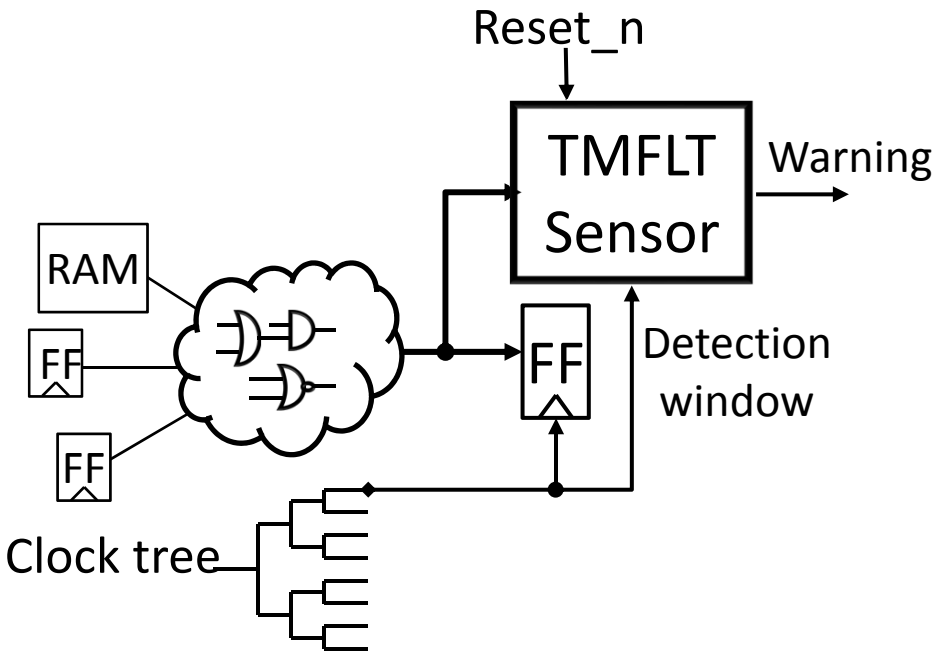
- Correlation between $F_{MAX}$ and frequency predictions
  - 5 slots into 21 dies @ 1V & 25°C (**+4/-3% accuracy**)
  - 1 slot into 21 dies @ [0.8V-1.3V] & 25°C (**6% accuracy**)
  - Not intrusive / Need complementary solution in UWVR



# $F_{MAX}$ tracking in UWVR : TMFLT

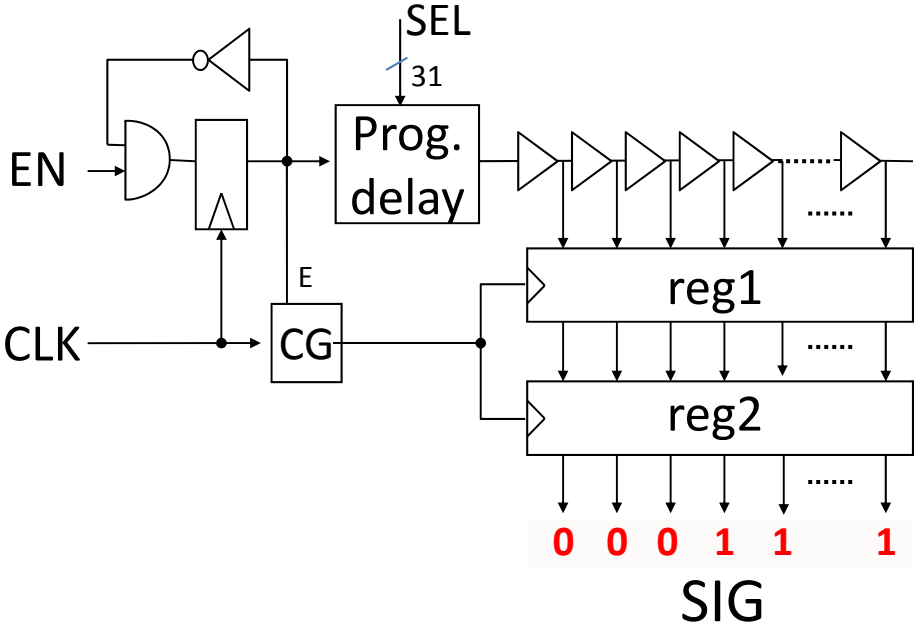
## **128** TMFLT-Sensors

- Analyze the registers slack time
- 300ps detection window




## **1** TMFLT-Ring

- Programmable replica path
- Time-to-digital measurement



# $F_{MAX}$ tracking in UWVR : TMFLT

## **128** TMFLT-Sensors

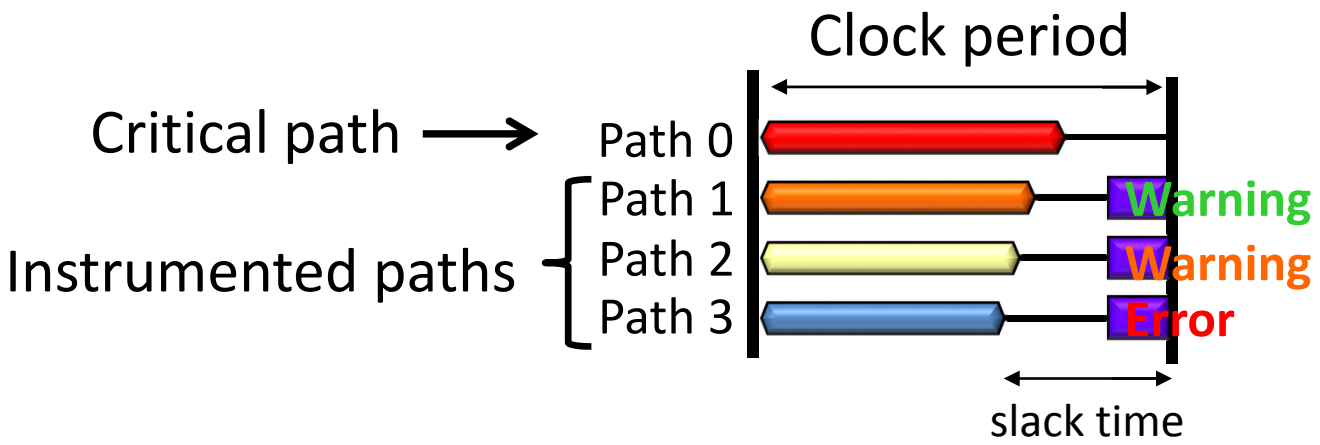
- Instrumented data path are **not** necessarily critical

⇨Only activated at calibration

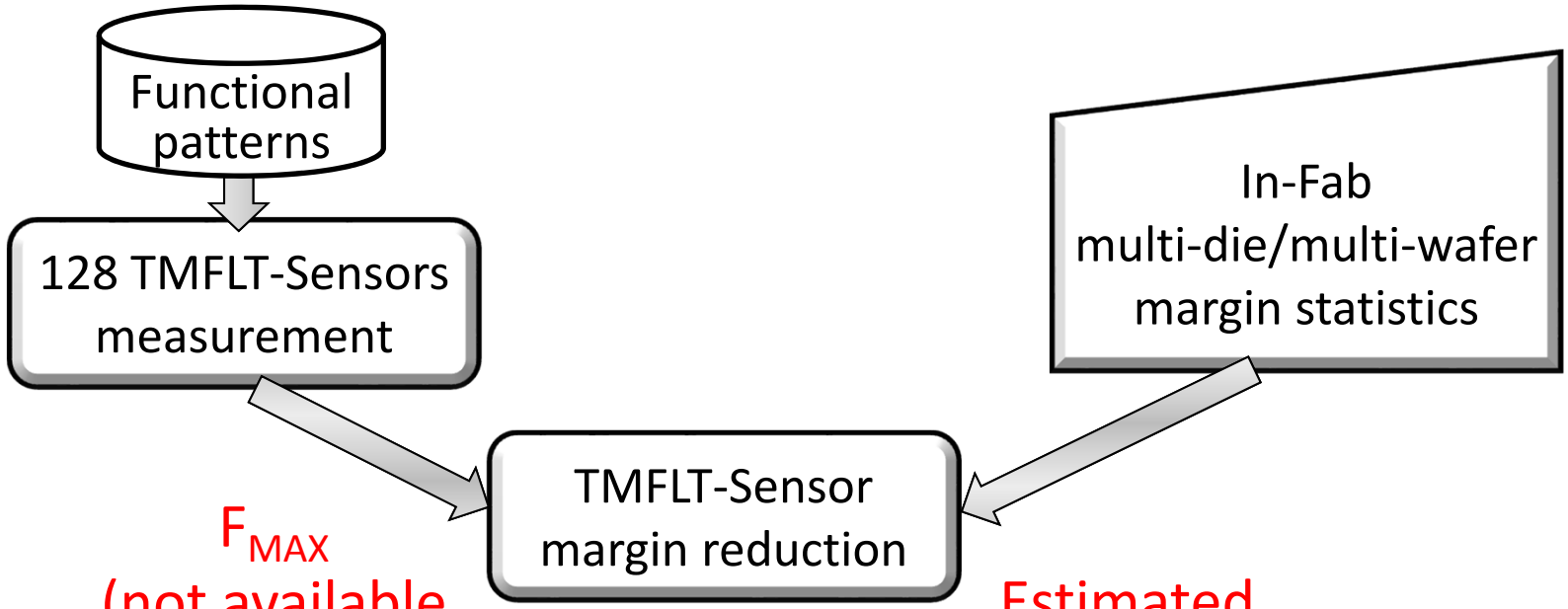
## **1** TMFLT-Ring

- Programmed with user IR drop margins

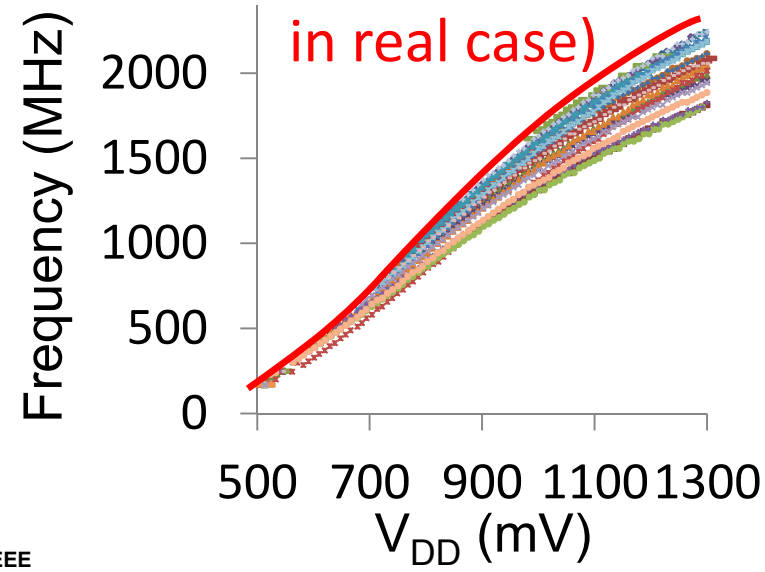
⇨ Activated at runtime



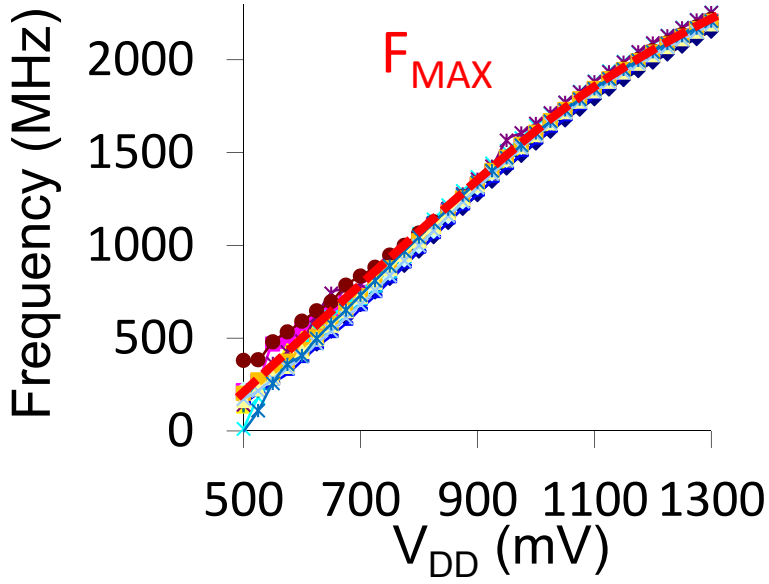
# $F_{MAX}$ tracking: TMFLT-S calibration



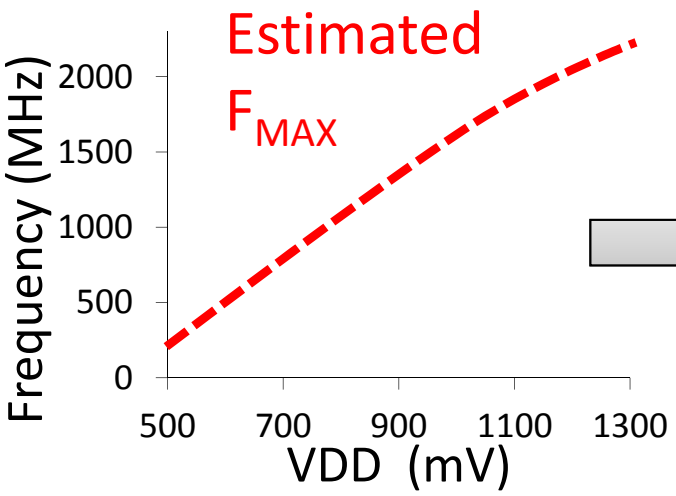

$F_{MAX}$ (not available in real case)

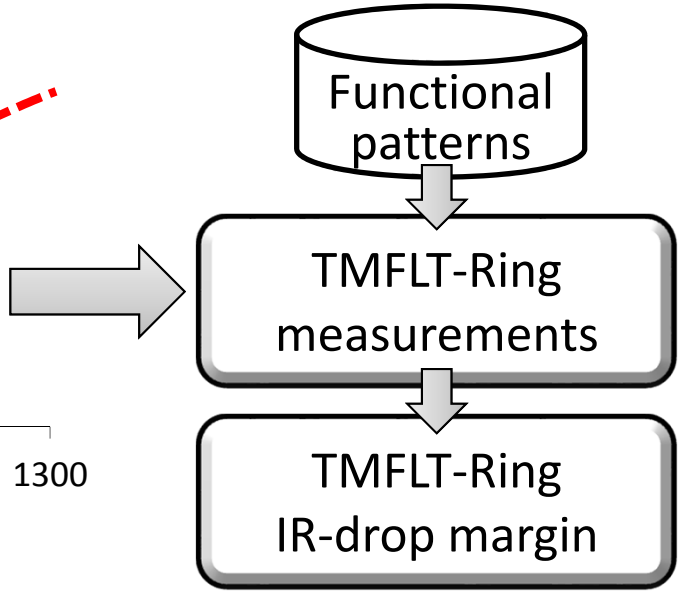


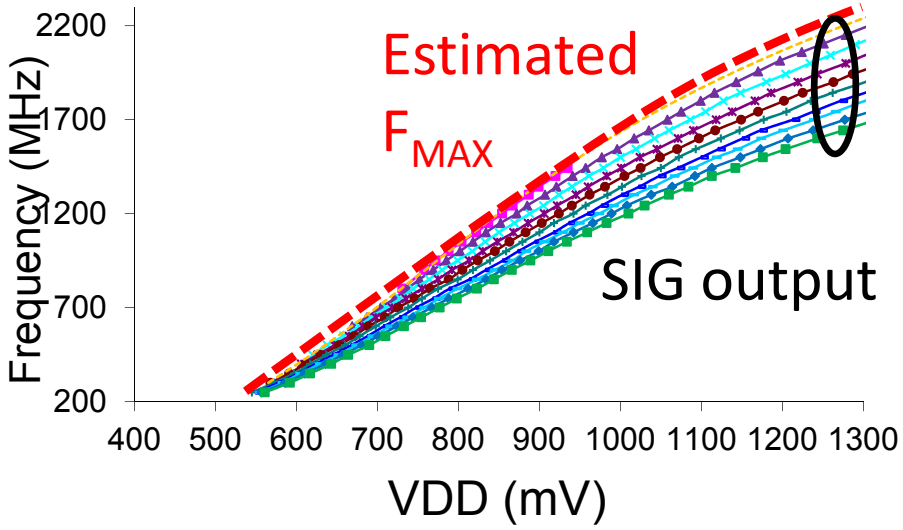

Estimated $F_{MAX}$

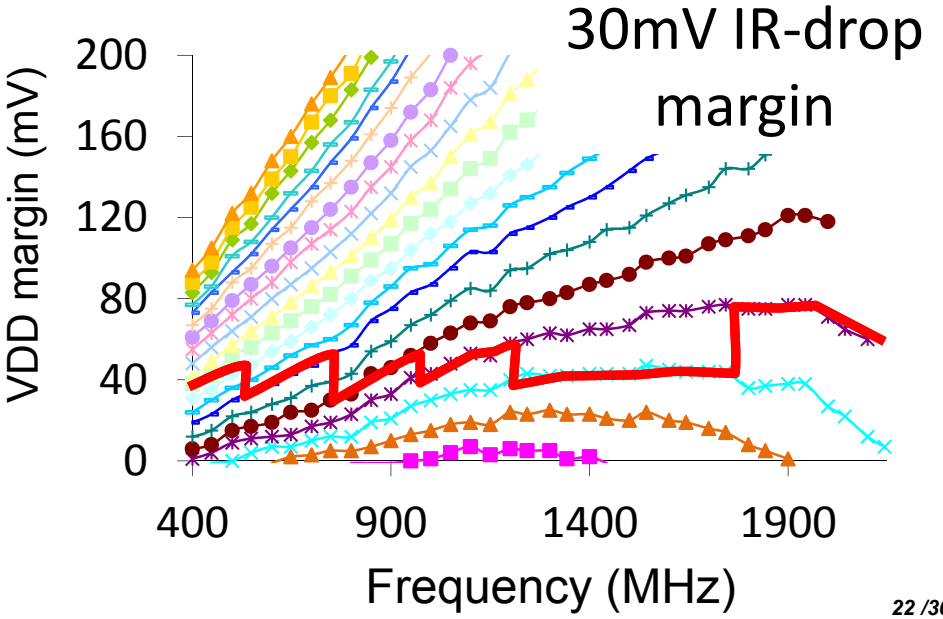


# $F_{MAX}$ tracking: TMFLT-R calibration

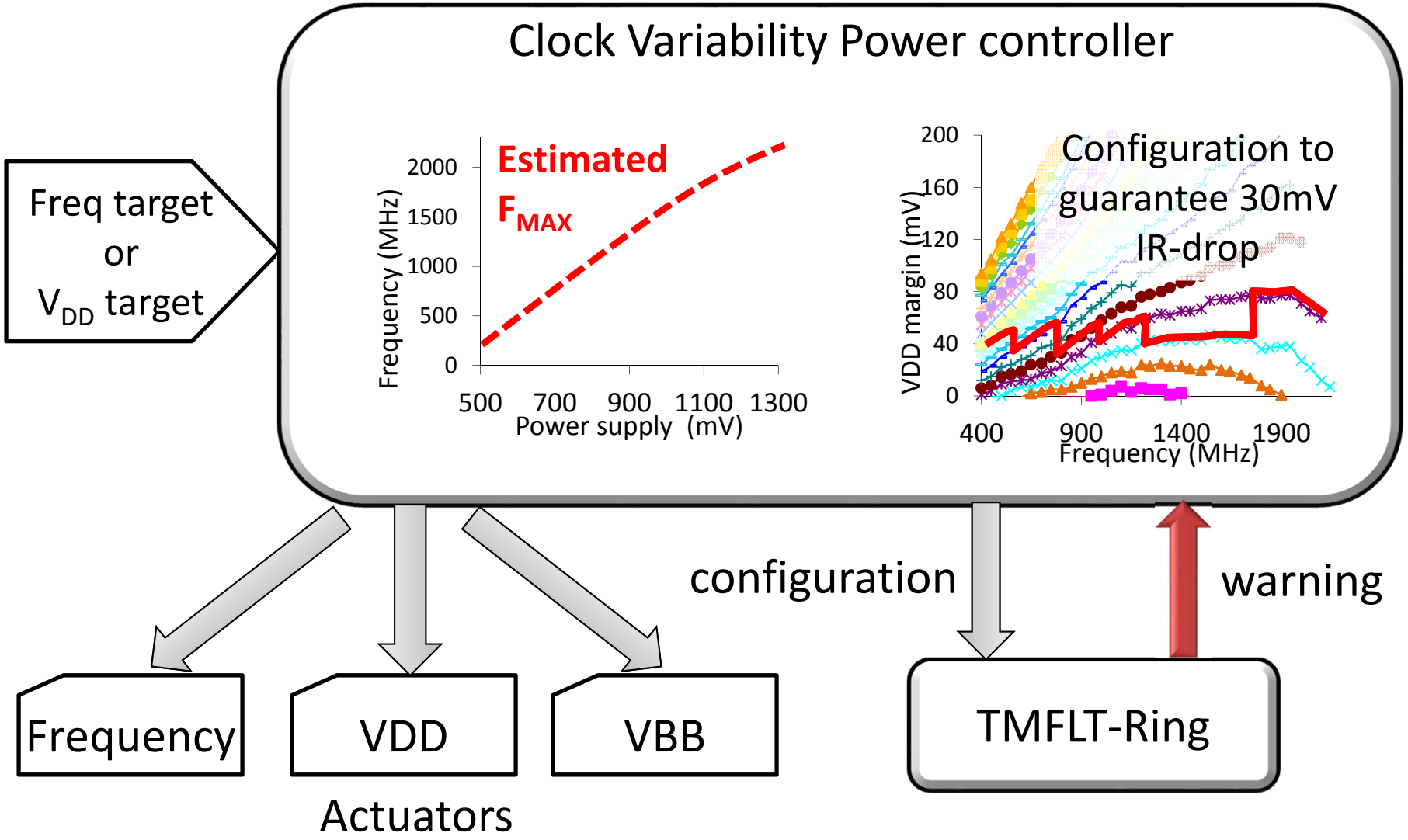


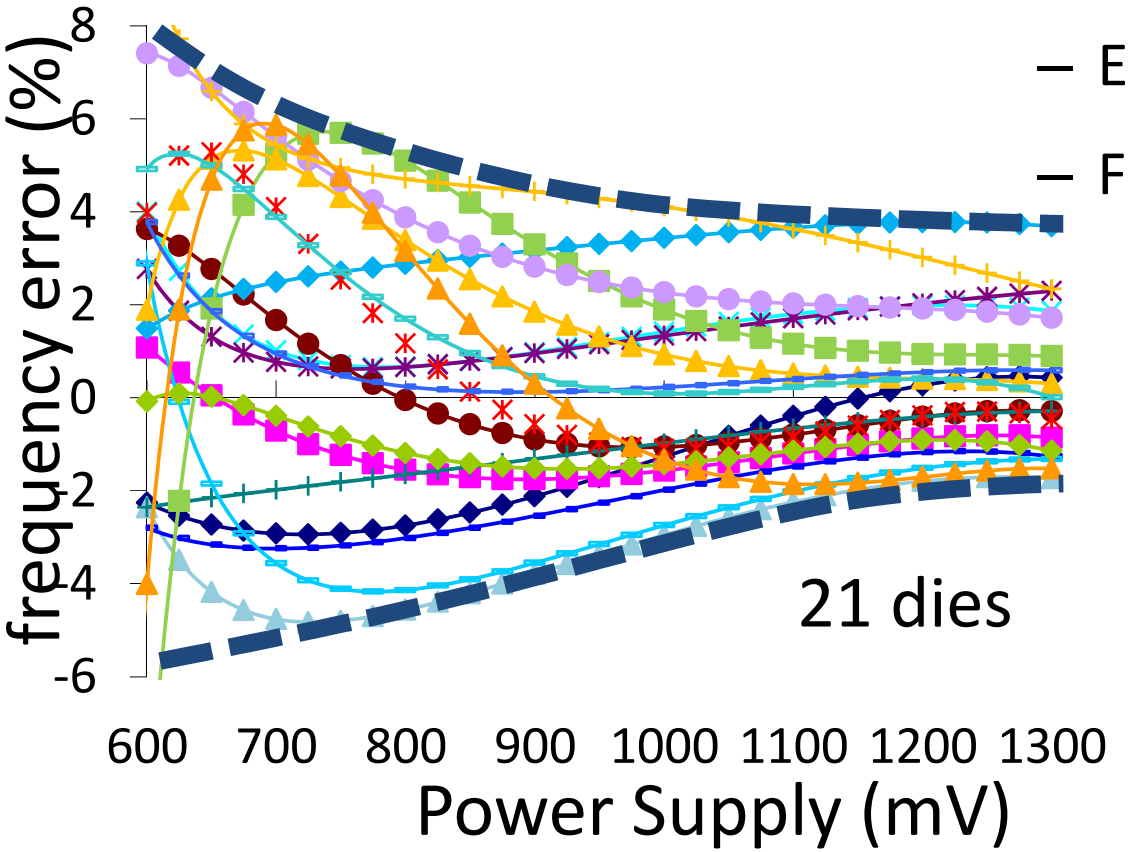



# $F_{MAX}$ tracking : TMFLT at runtime



# TMFLT results

- Frequency estimation error compared to $F_{MAX}$
  - **Error of ±4% at 1V**



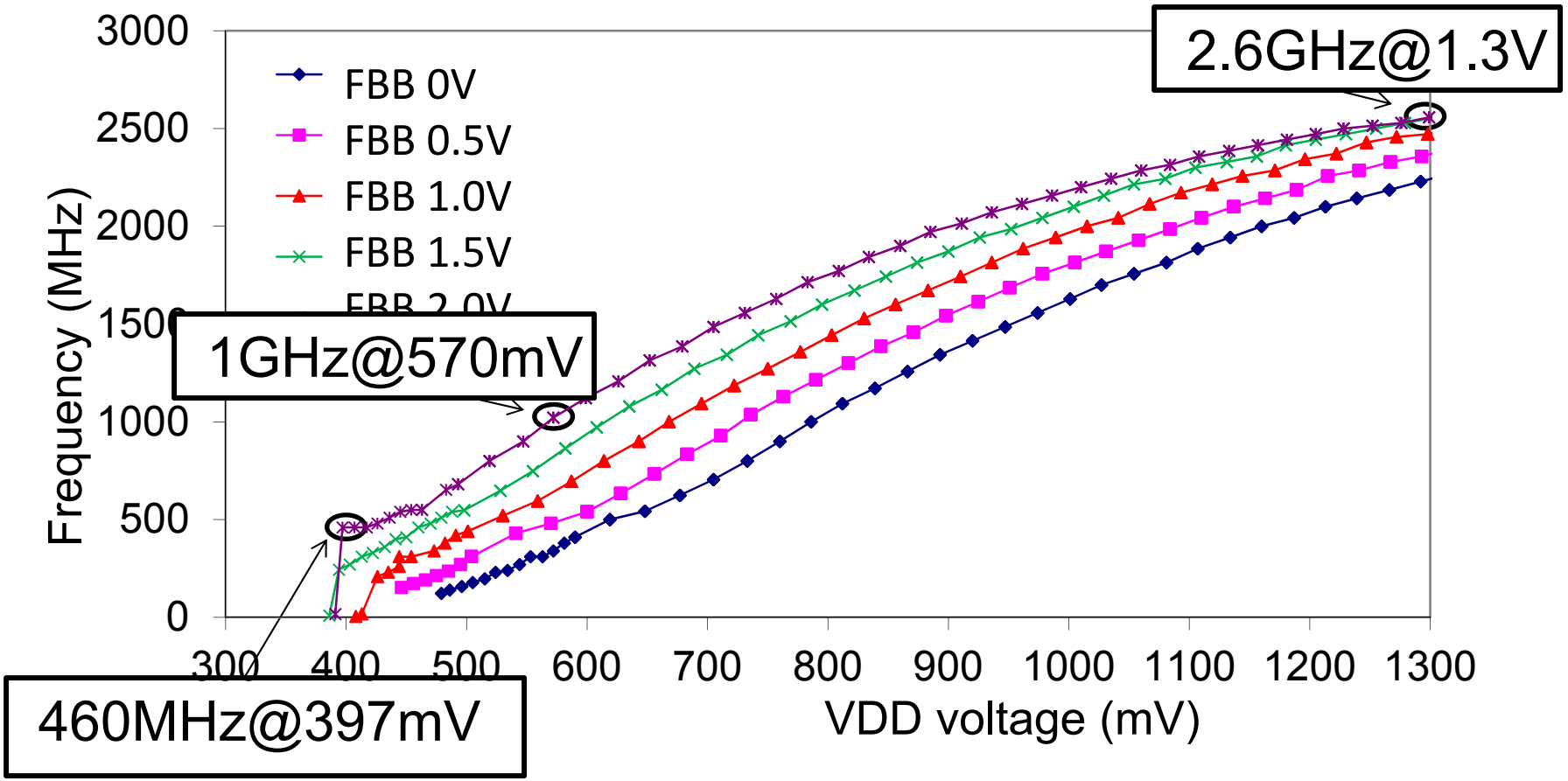

- Compared to worst case PVT corner (3σ) approach without $F_{MAX}$ tracking:
  - Energy gain of 40.6% @ 0.6V
  - Frequency gain of 24% @1V

# Outline

## Achieving Performance in **UWVR**

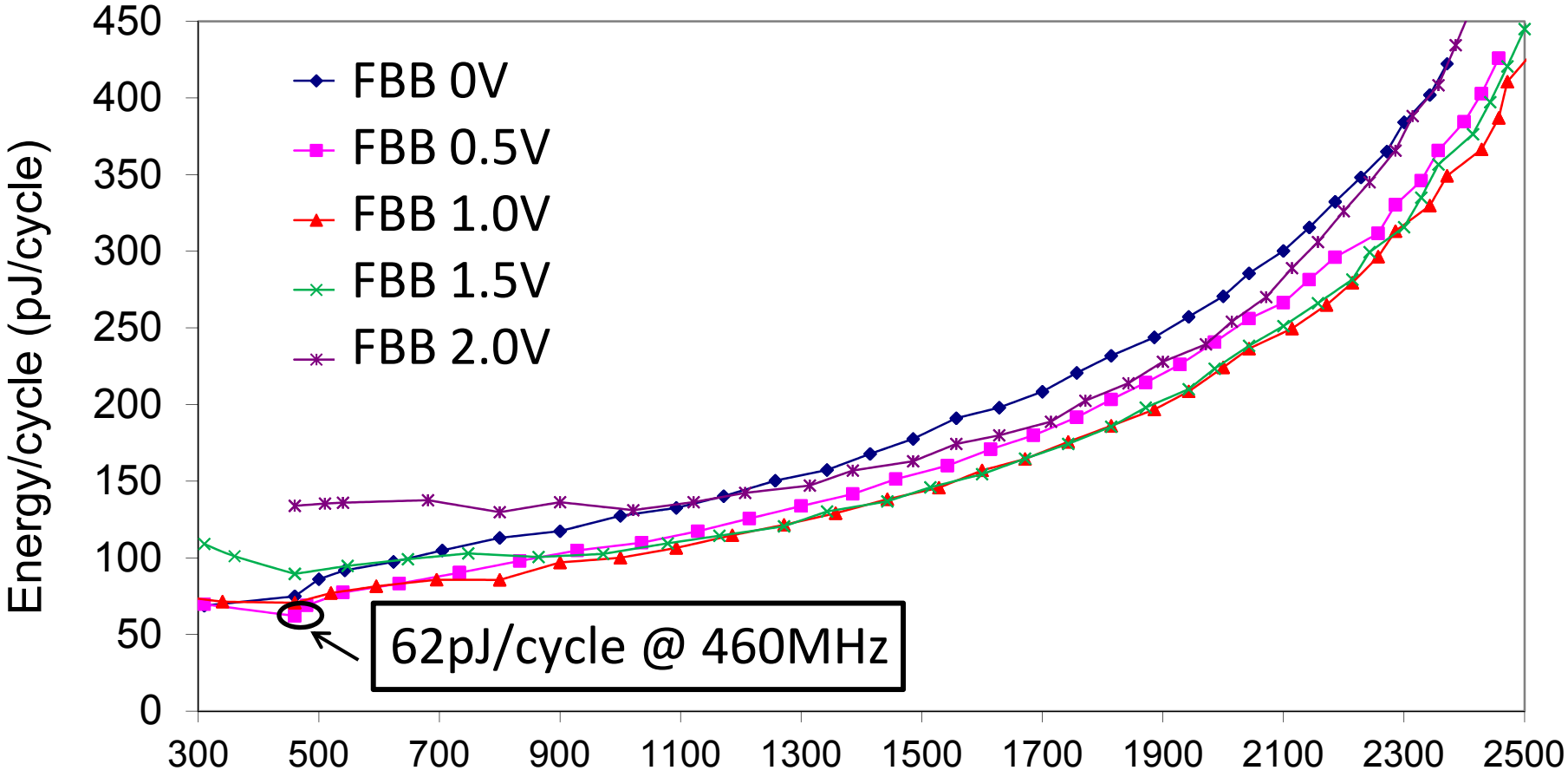
- UTBB FDSOI 28 nm technology
  - Well-type methodology
  - Body Biasing techniques
- Optimized library sub-set
  - Assymetric standard cells
  - High performance flip-flops
- $F_{MAX}$ tracking techniques
  - Replica path and Timing fault
- DSP performances silicon results

# DSP speed performance measured results



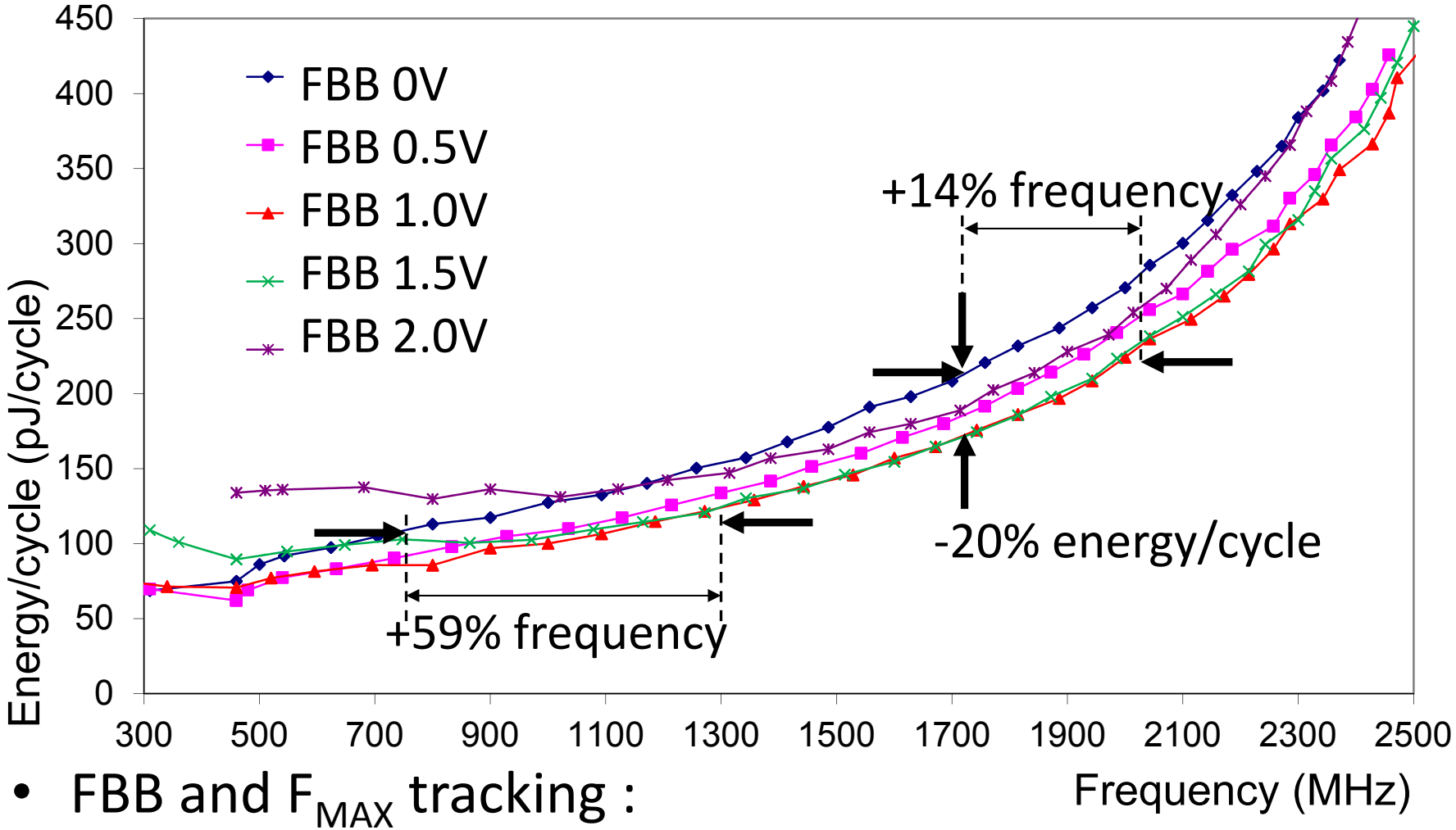

- FBB up to +2V and $F_{MAX}$ tracking :
  - ↗ frequency up to 460MHz at minimum voltage

# DSP energy performance measured results



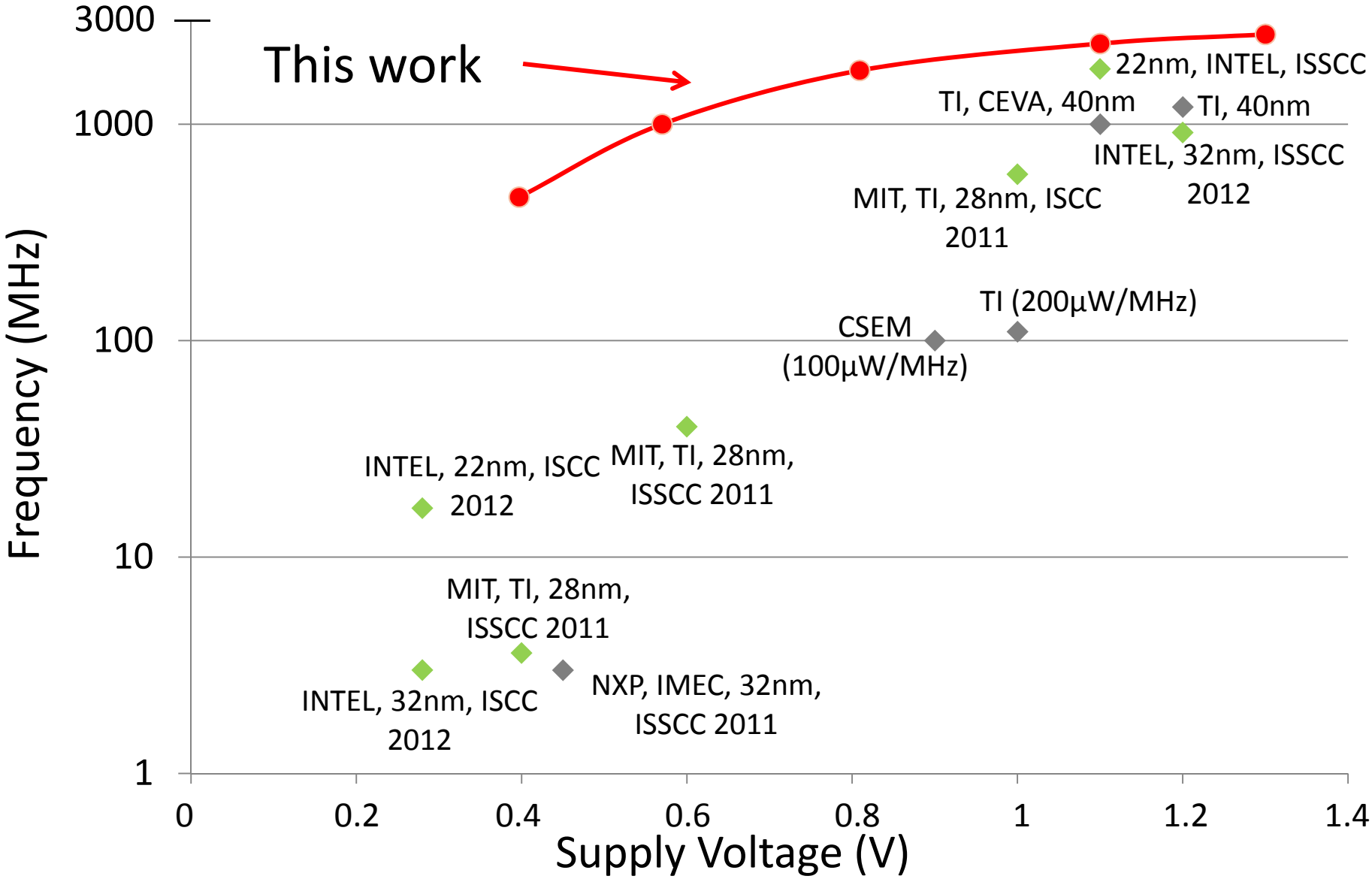

- For a fixed voltage supply :
  - Lowest energy at 460MHz
  - Power consumption : 370mW at 1V

# DSP energy performance measured results




- FBB and $F_{MAX}$ tracking :
  - ↗ Frequency up to 59% (target 100pJ/cycle)
  - ↘ Energy by 20% (target 1.7 GHz)

# Comparison with State of the Art WVR



# Conclusion

- Very **high speed** and **low energy** 28 nm DSP achieved by the efficient use of $F_{MAX}$ tracking techniques combined with optimized libraries in UTBB FDSOI technology
  - Demonstrated almost 2 decades higher operating frequency at 0.4V… while maintaining State of the Art high speed operation
- **Convergence** between High Performance and Low Power is possible by using innovative **Ultra Wide Voltage Range** design techniques
- New features for increased **Versatility** in future integrated circuits

# A 6mW 5K-Word Real-Time Speech Recognizer Using WFST Models

Michael Price, James Glass, Anantha Chandrakasan
MIT, Cambridge, MA

# Vision

- Allow speech interfaces to be added to any electronic device
- Use scalable architecture, so the application can dictate tradeoffs
  - Available energy/power
  - Accuracy of recognition
  - Complexity of model (e.g. vocabulary, grammar)
- Focus on low power first
  - Baseline design: 5,000 words (Wall Street Journal dataset)
  - Real-time; ~10 mW power
  - Constrained memory size and bandwidth

**Potential applications**



Wearable device

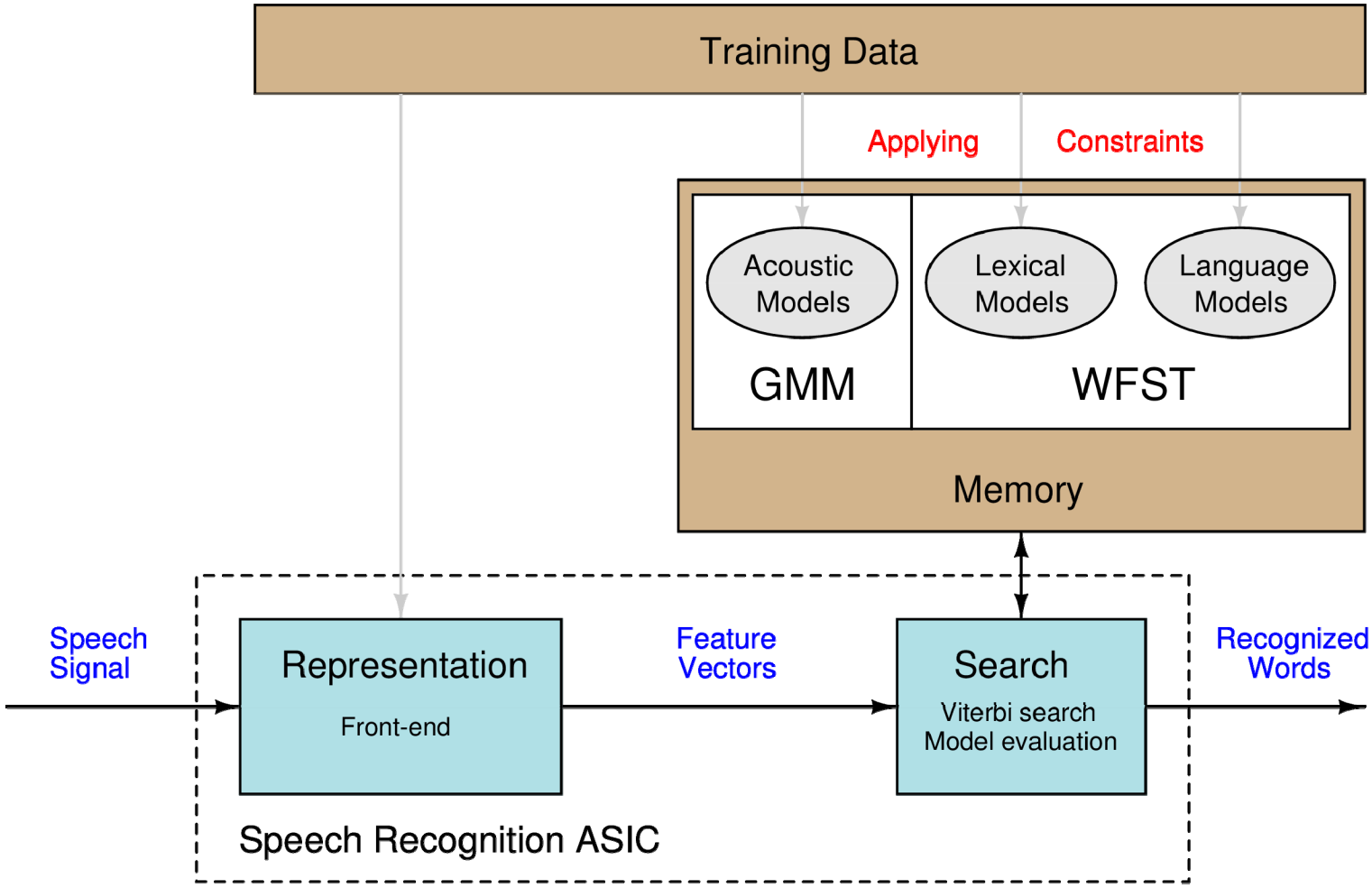

Smartphone


Appliance

# Presentation Outline

- Target devices and applications
- Speech recognition system design
- Hardware decoder architecture
- Architectural enhancements
- Test chip results

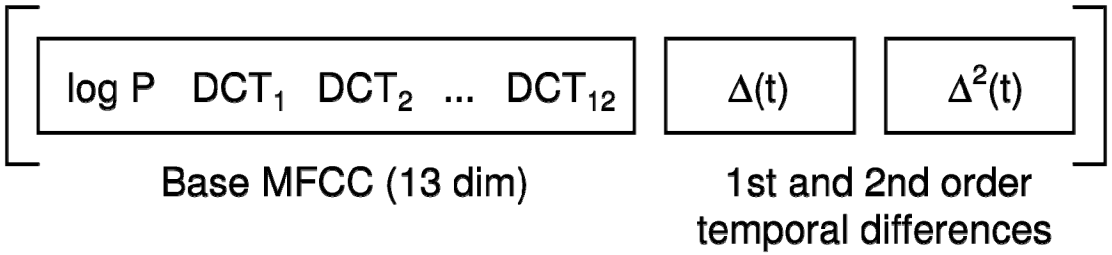
# How Speech Recognition Works




- Training (software): organizes knowledge about expected speech
- Decoding (hardware): develops hypotheses about received speech

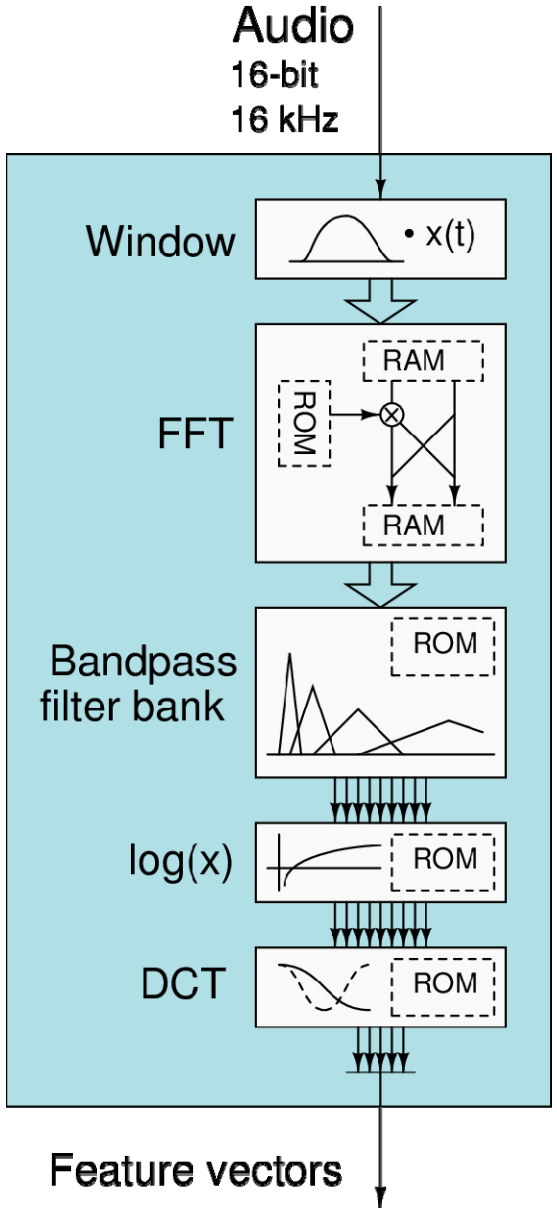
# Front-end: Computing Feature Vectors

- Reduces sample rate from 16 kHz to 100 Hz
  - 39-dimension feature space

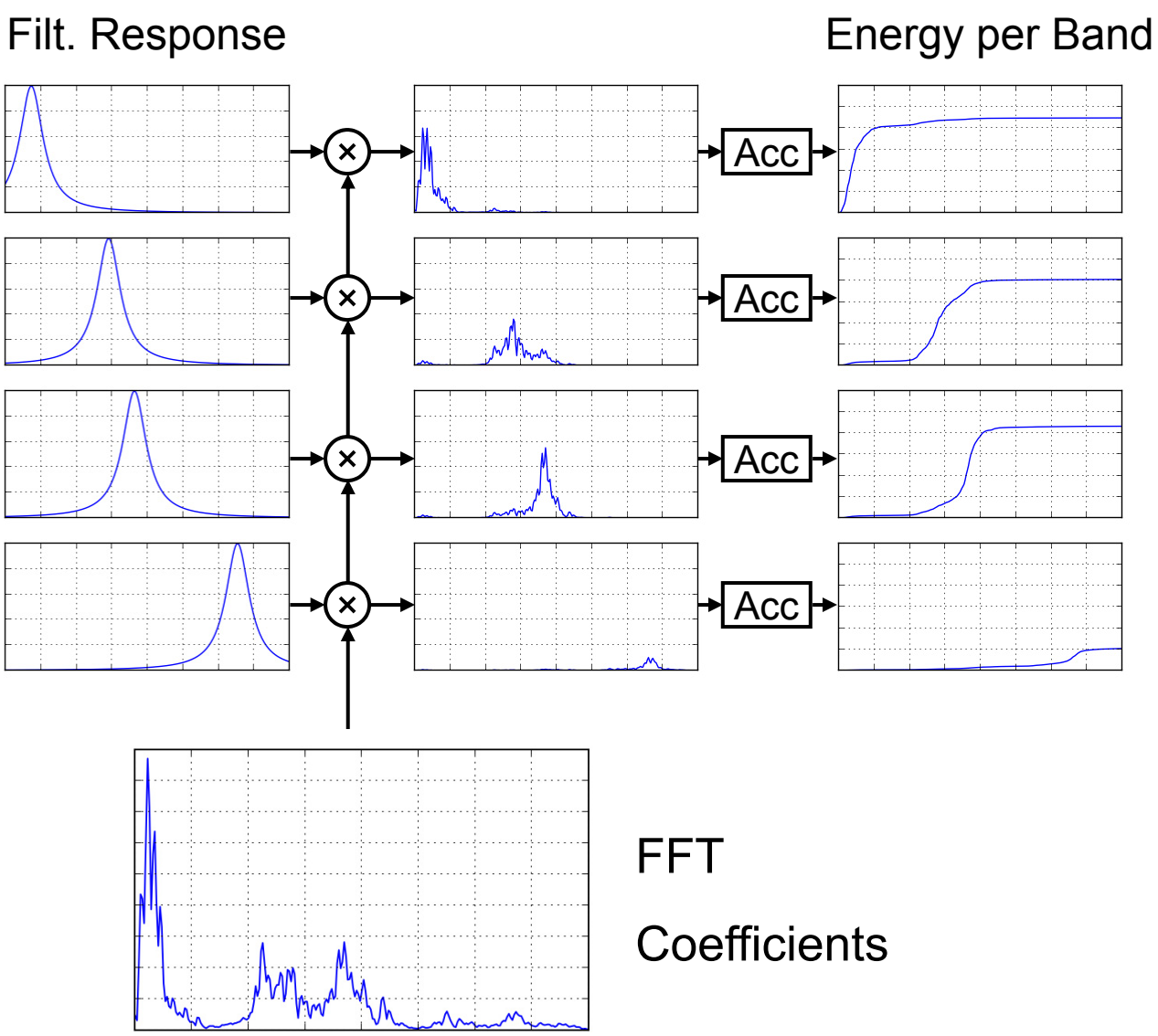


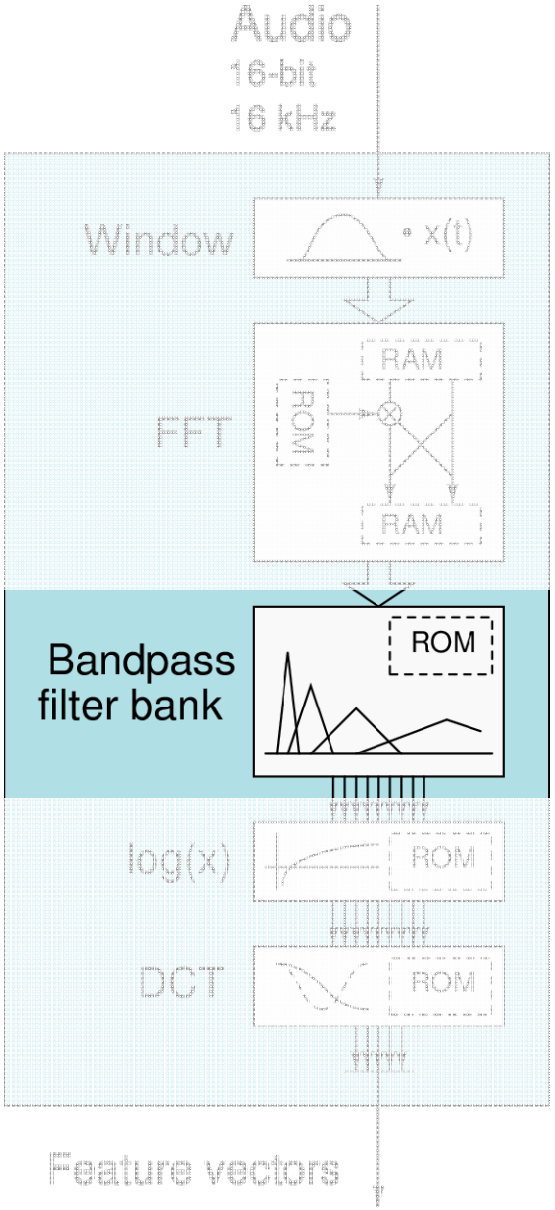

- Extracts important information
  - Frequency scale warped to match human perception
  - Convolution of waveform = addition in feature space

- Computationally "easy"
  - All parameters fit in on-chip memory
  - Reduced clock frequency (min. 600 kHz)



# Front-end: Conventional Bandpass Filter

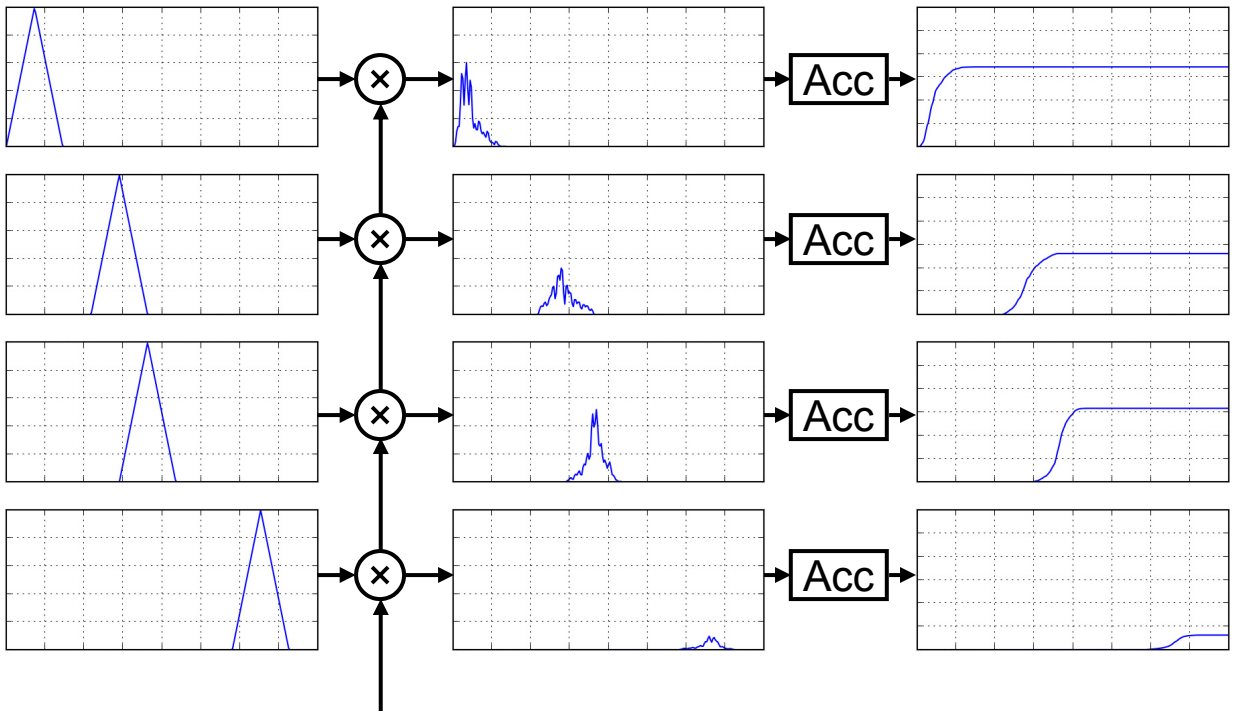


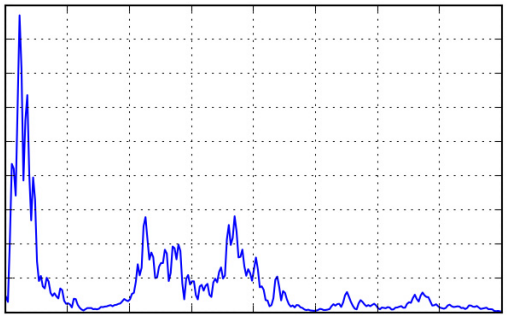



# Front-end: Triangular Bandpass Filter

Filt. Response (Triangular)

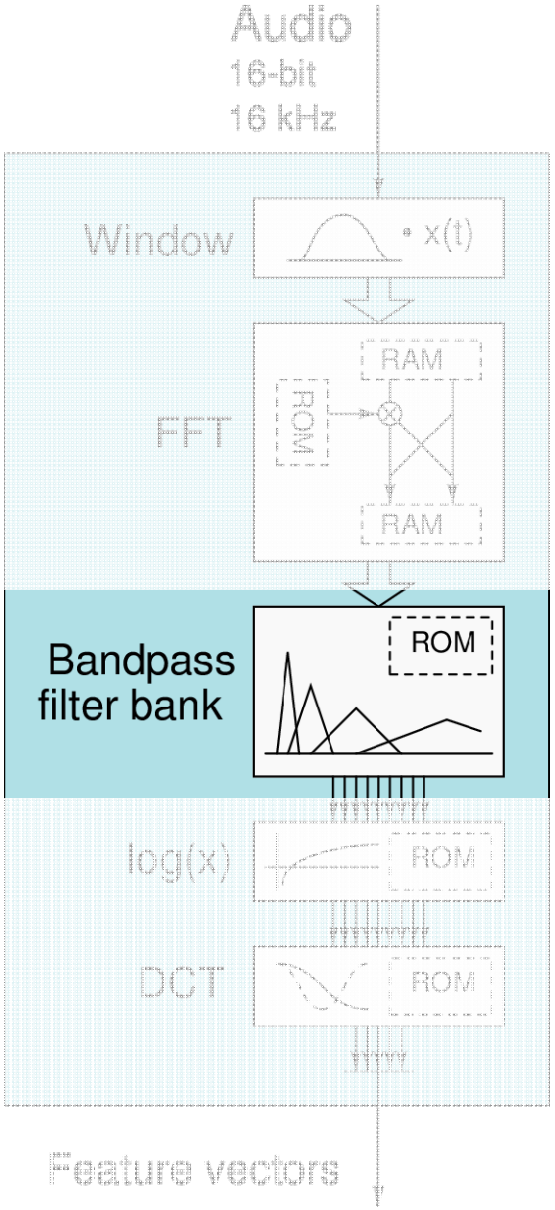
Energy per Band



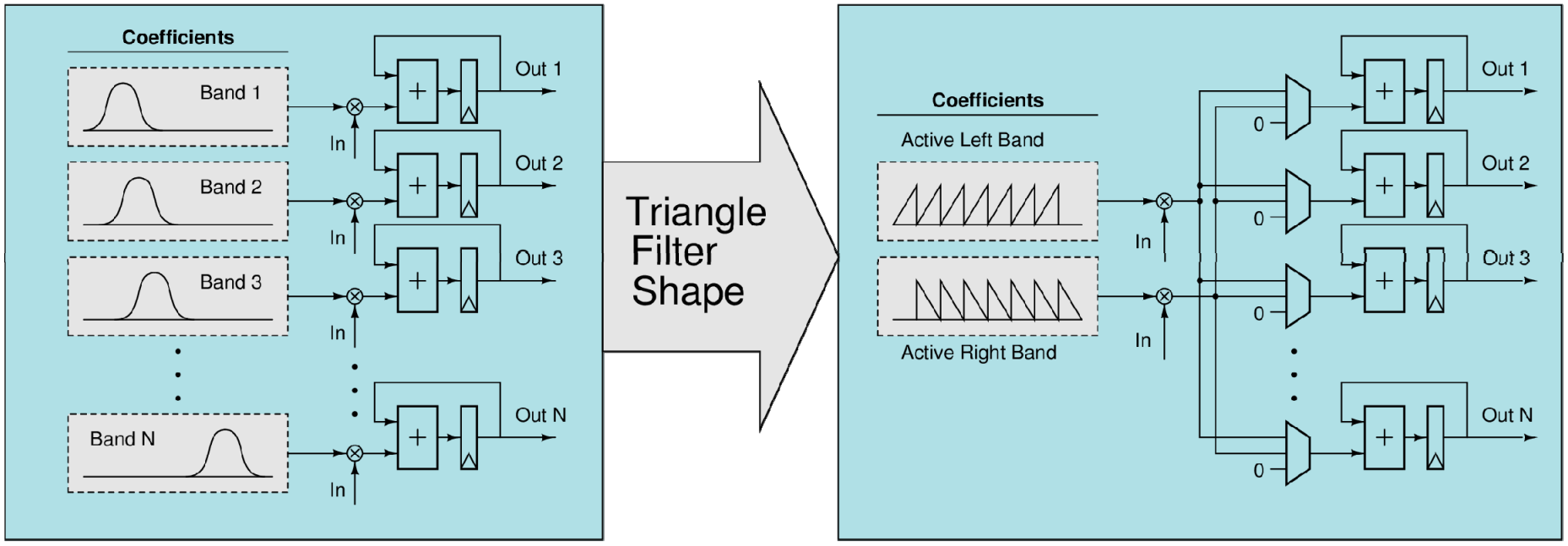



FFT

Coefficients



# Front-end: Triangular Bandpass Filter

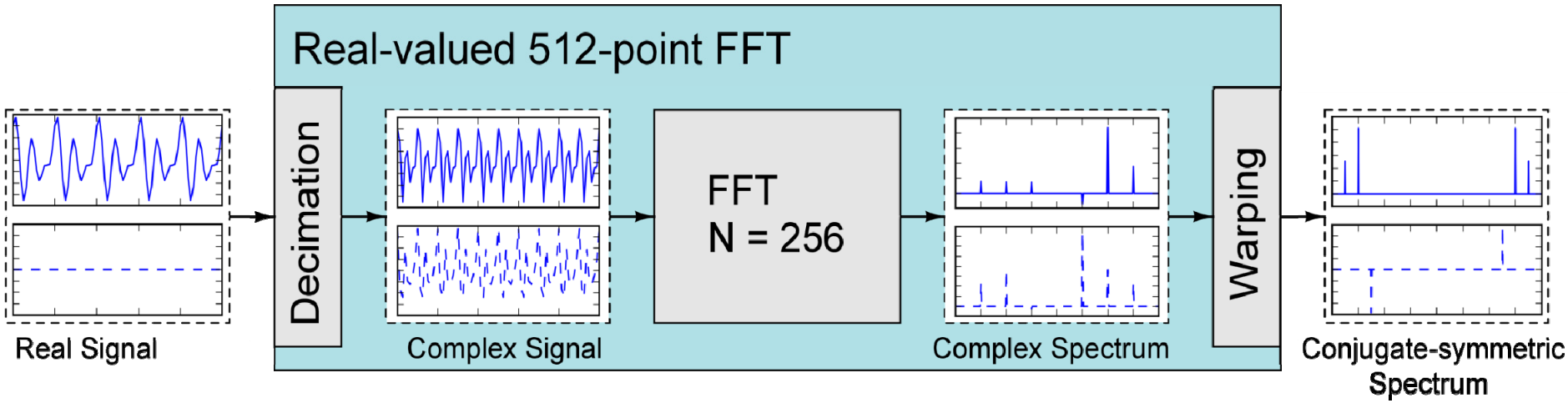



N=26 multipliers

2 multipliers

**Reduces area of bandpass filter bank**

# Front-end: Real valued FFT



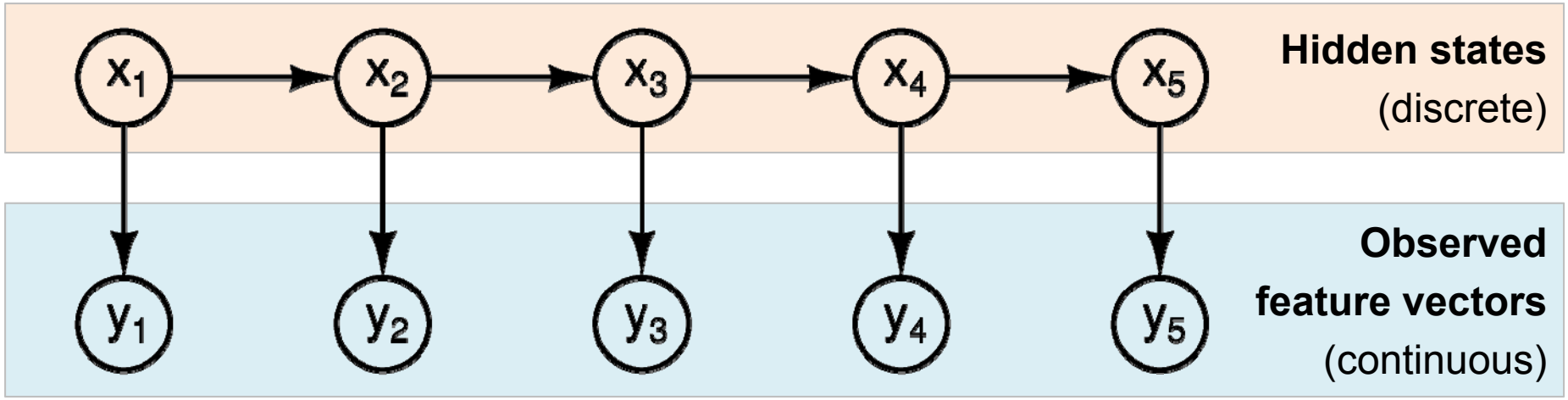

**Allows clock frequency to be cut in half**

Result: 600 kHz, 110 μW (avg.)

# Hidden Markov Model (HMM) Framework

- Assume speech sounds (feature vectors) are generated by an underlying random process
- As feature vectors arrive, maintain and score hypotheses for state trajectories

**Statistical model of speech**

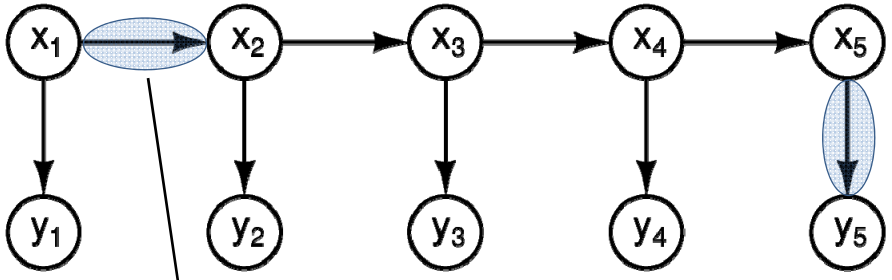


# Hidden Markov Model (HMM) Framework

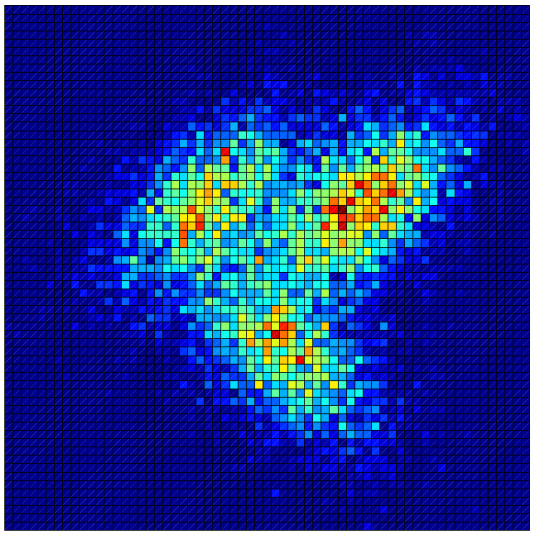
- Assume speech sounds (feature vectors) are generated by an underlying random process
- As feature vectors arrive, maintain and score hypotheses for state trajectories

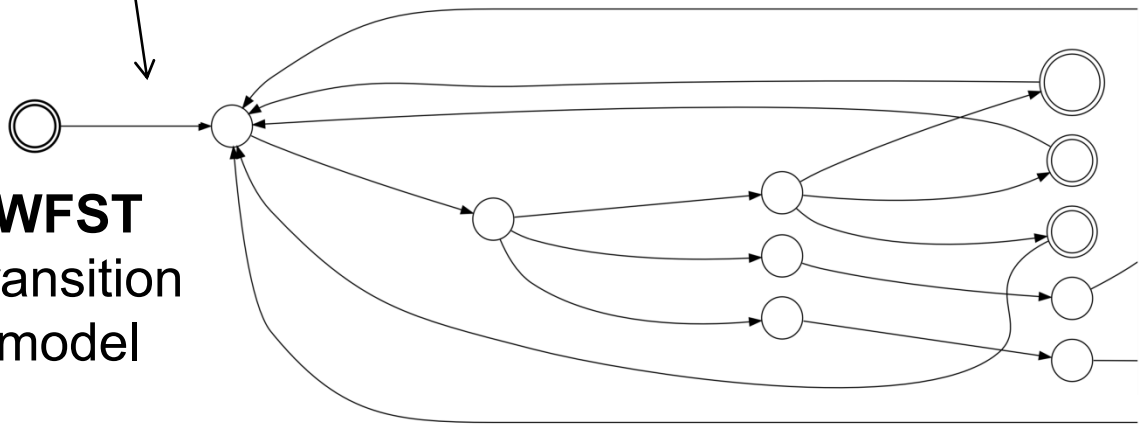
**Statistical model of speech**

Hidden Markov model (**HMM**)




**GMM**
Emission model





**WFST**
Transition model

# Acoustic Model: GMM

## Represents $p(y_t \mid x_t)$ in HMM

Probability of feature vector,

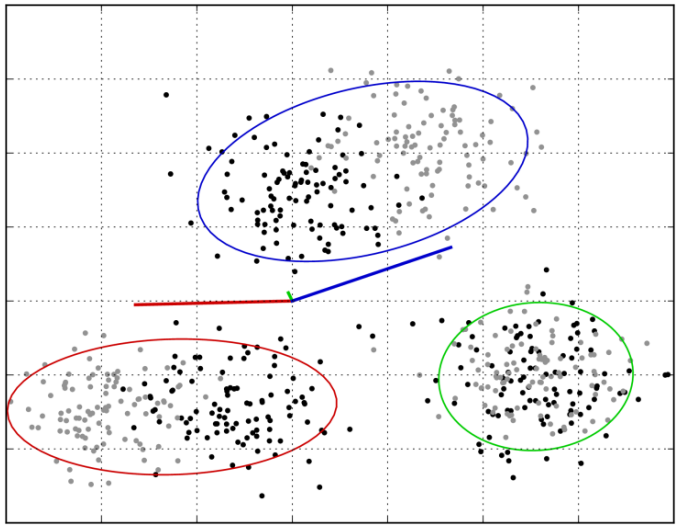
conditioned on the current WFST state

## Properties

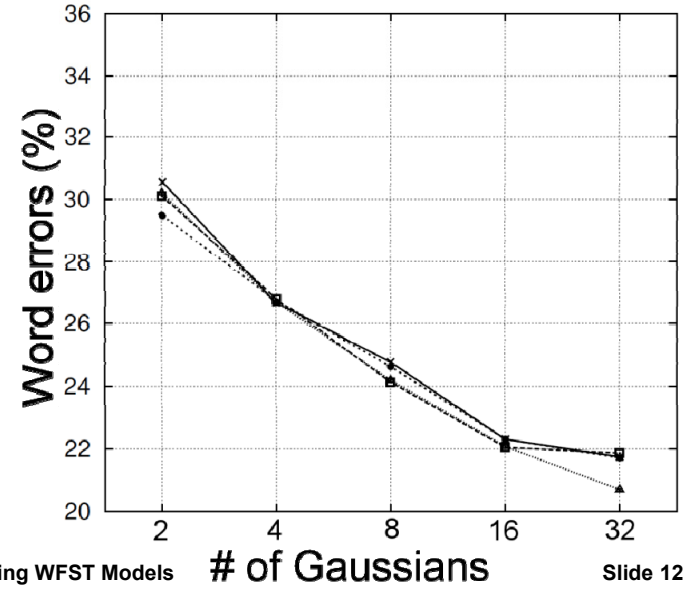
- Continuous distribution
- Diagonal covariance (for simplicity)
- Easy to evaluate

**Parameters (e.g. mean and variance) can demand high memory bandwidth**

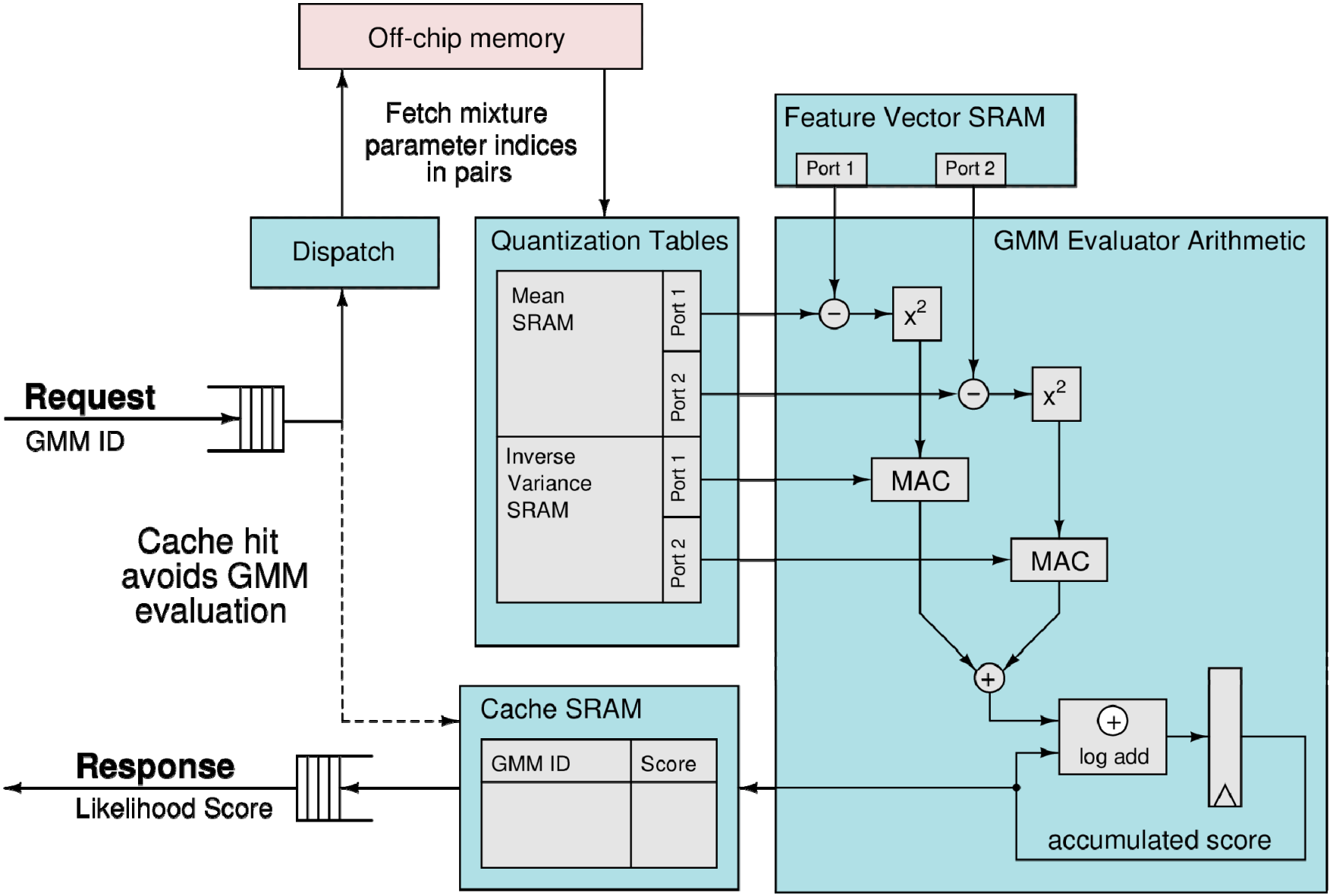
**A 2-D example (3 components)**



**Accuracy improves with additional components**

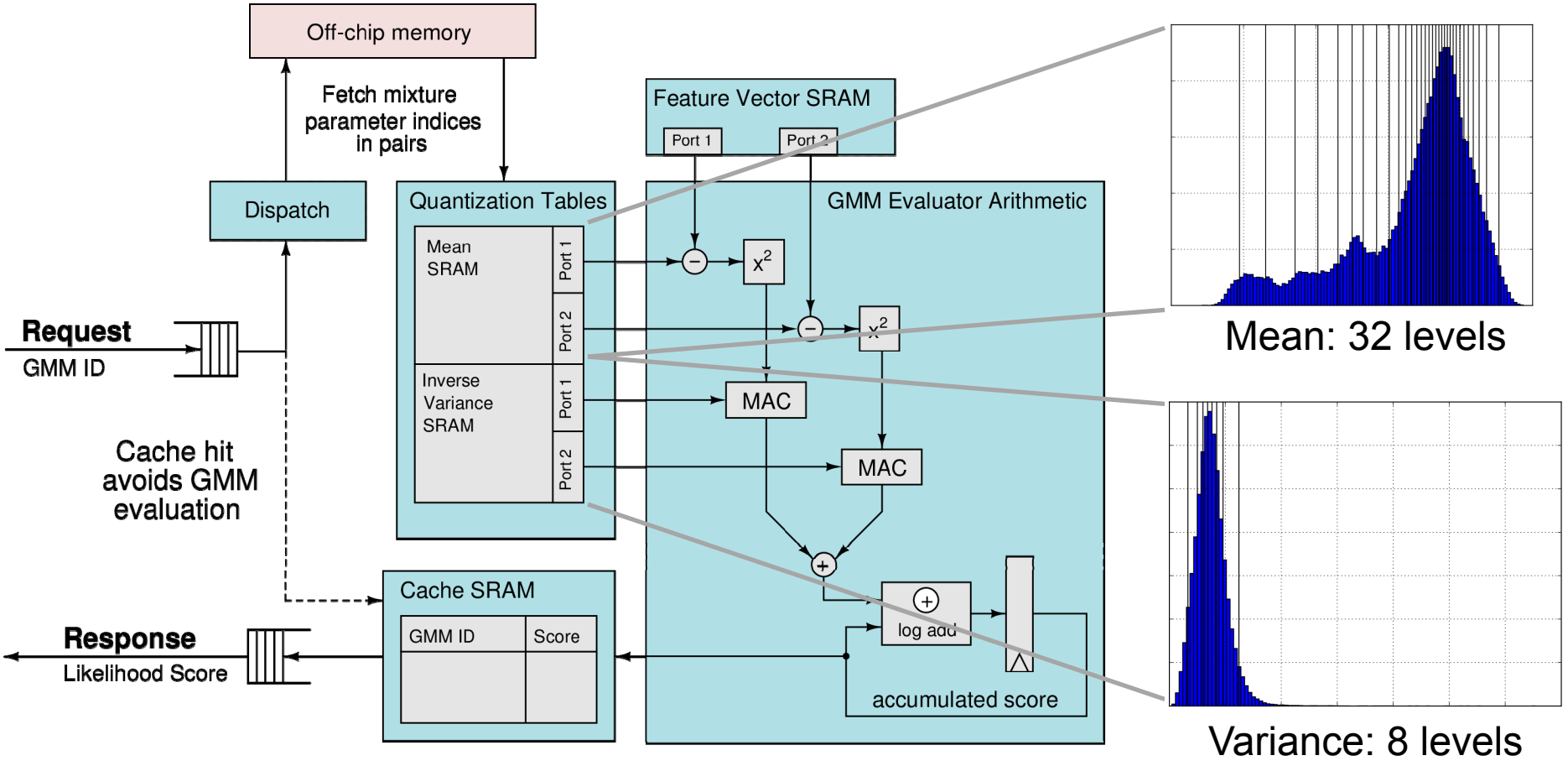


# GMM: Architecture and Caching



# GMM: Parameter Quantization

| # GMM parameters | 10.2M |
|---|---|
| Mem. bandwidth - Naive | 2964 MB/s |
| Mem. bandwidth - Cached | 389.9 MB/s |
| Mem. bandwidth - Quantized | 53.72 MB/s |



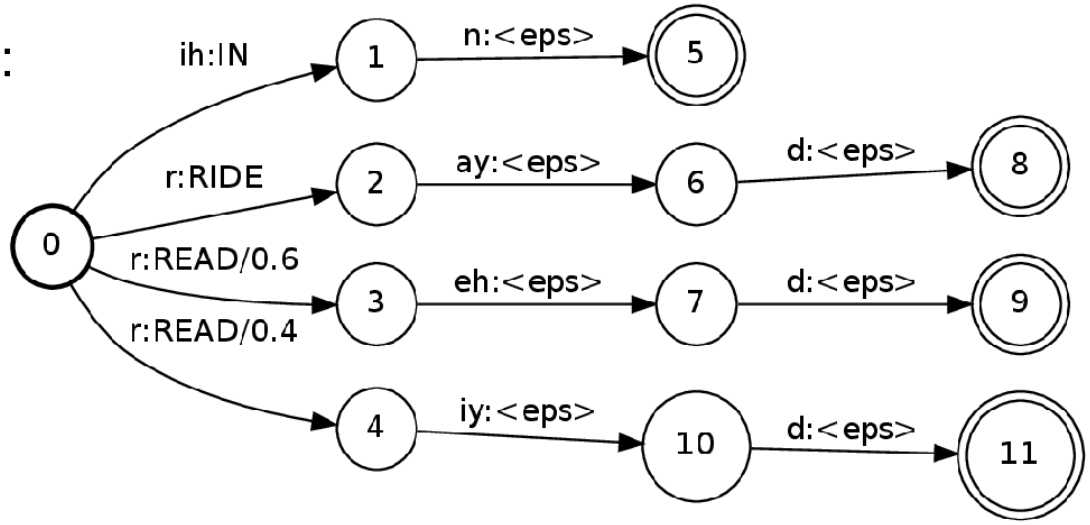
# Transition Probability Model: WFST

**WFST**: Weighted finite-state transducer

Developed by Mehryar Mohri et al., AT&T Labs, mid-1990s

- A state machine whose transitions (arcs) encode:
  - An input label
  - An output label
  - A weight (e.g. probability)

**Example: pronunciation model**




- Each WFST:
  - Allows only valid input sequences
  - Assigns a probabilistic “score” to each state sequence
  - Allows the word sequence to be computed from the state sequence

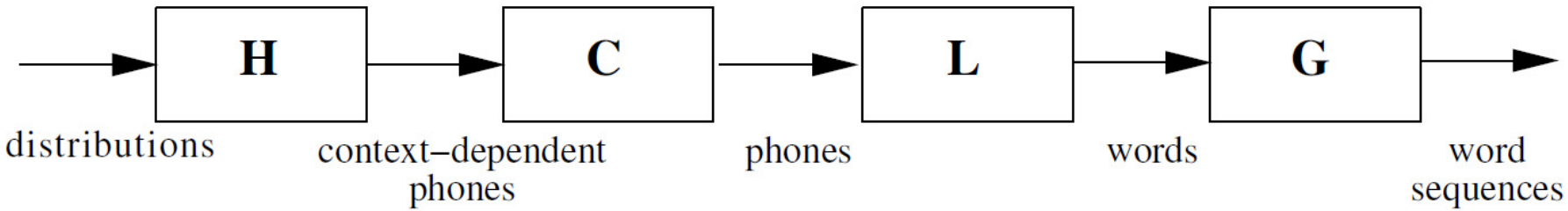
# Transition Probability Model: WFST

**WFST**: Weighted finite-state transducer

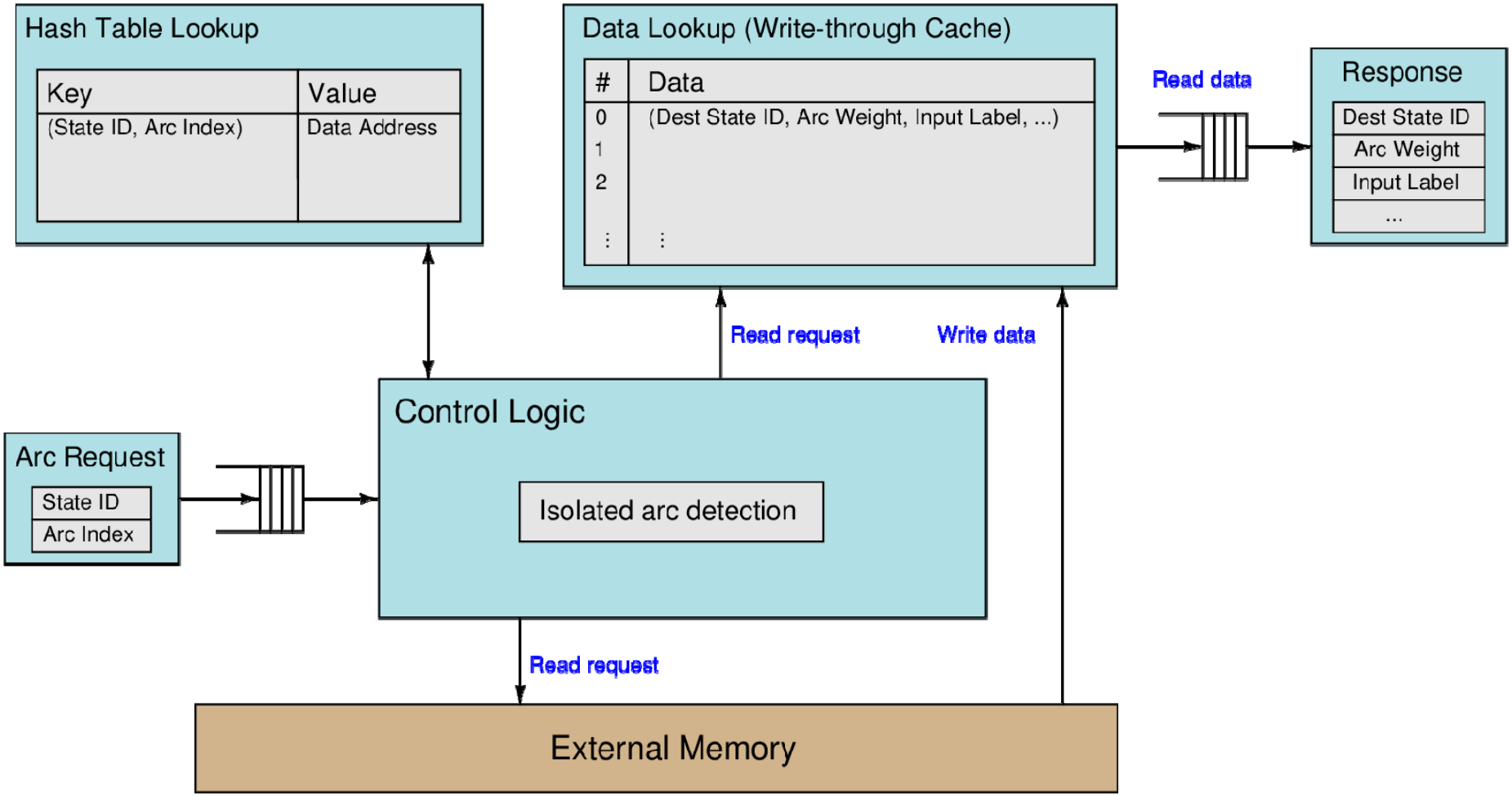
- Can represent multiple levels of modeling
  - H: subphonetic structure
  - C: phonetic context
  - L: valid pronunciations (lexicon)
  - G: word and N-gram probabilities (language model)

- Can be composed and statically optimized

**The speech recognition transducer cascade**



# WFST Arc Fetch Caching




| | Bandwidth | Pages/sec |
|---|---|---|
| Without cache | 8.01 MB/s | 55.4k |
| With cache | 5.73 MB/s | 18.0k |

**Fewer parameters, but sparse access**

# Search: Viterbi Algorithm

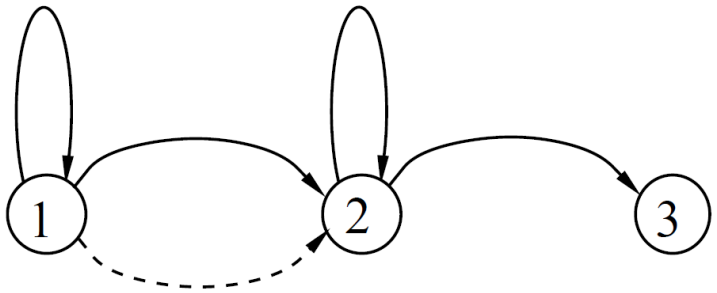
$$p(x_{t+1}) \approx \max_{x_t} p(x_t)\, p(x_{t+1}|x_t)\, p(y_{t+1}|x_{t+1})$$

WFST | GMM

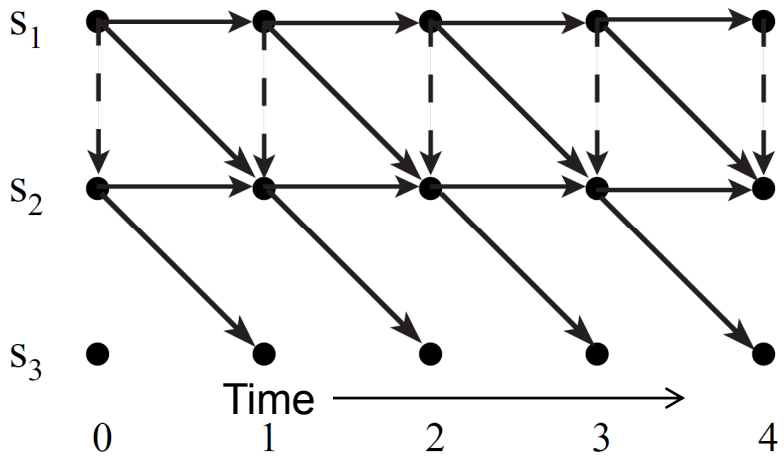
- Time synchronous (update after each frame)
- Maintain list of active states
- Save most likely destinations (with backpointer) in active state list

**Simplified example**

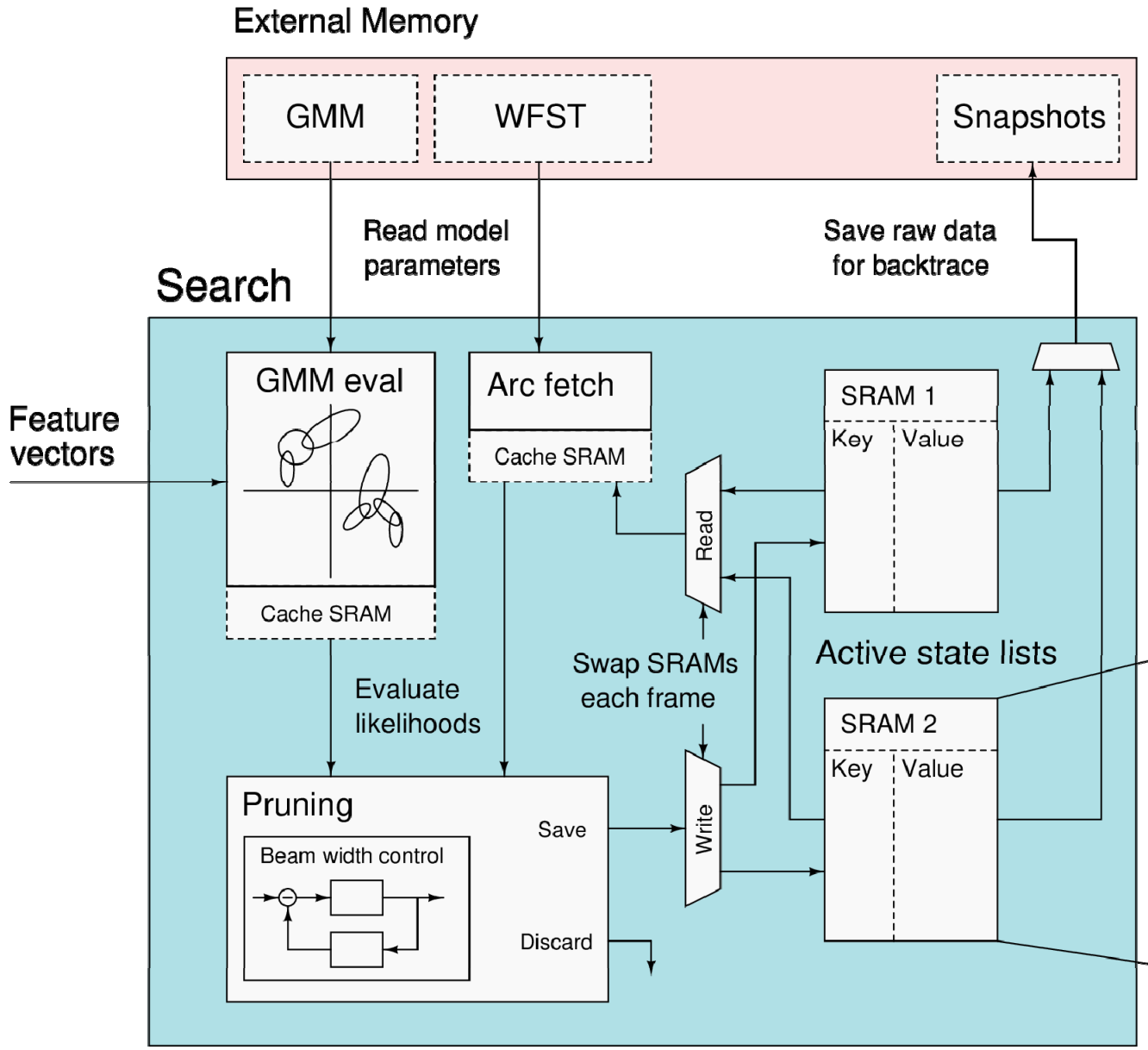
WFST model




Search trellis



# Search Architecture

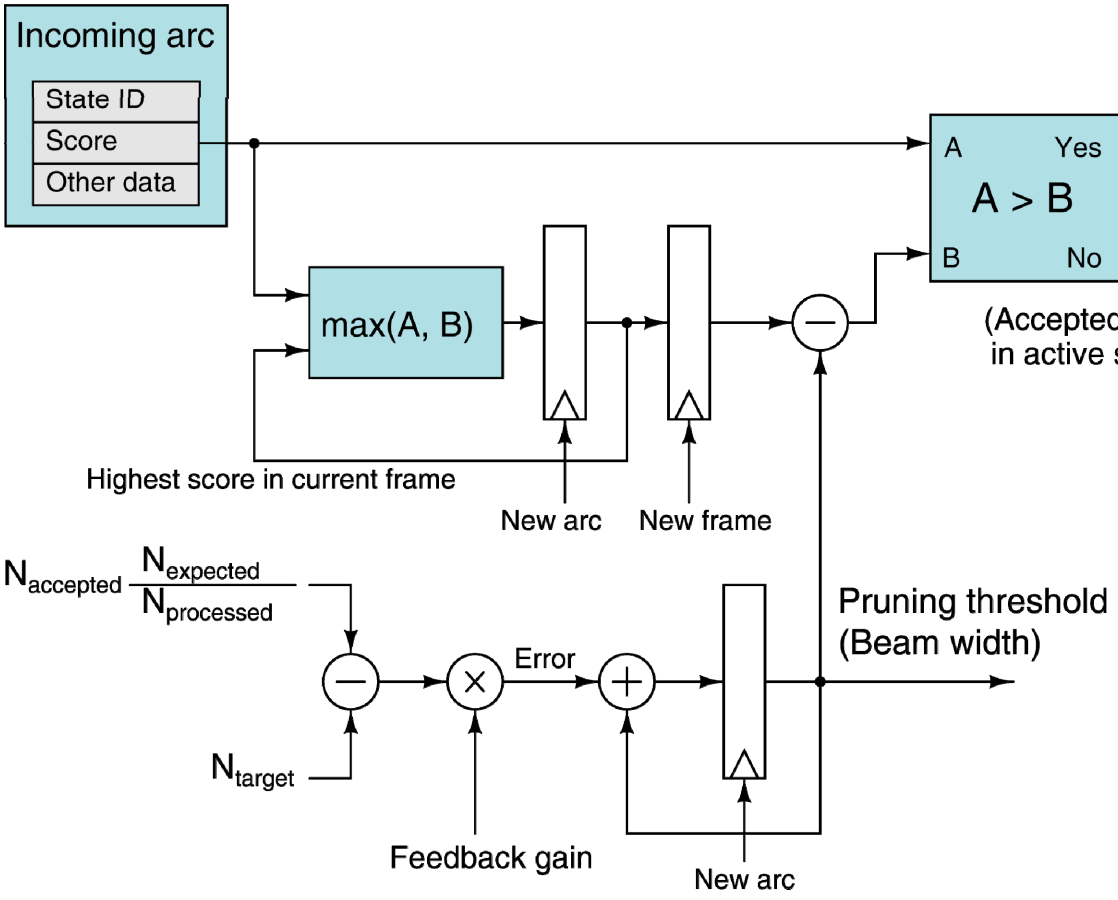


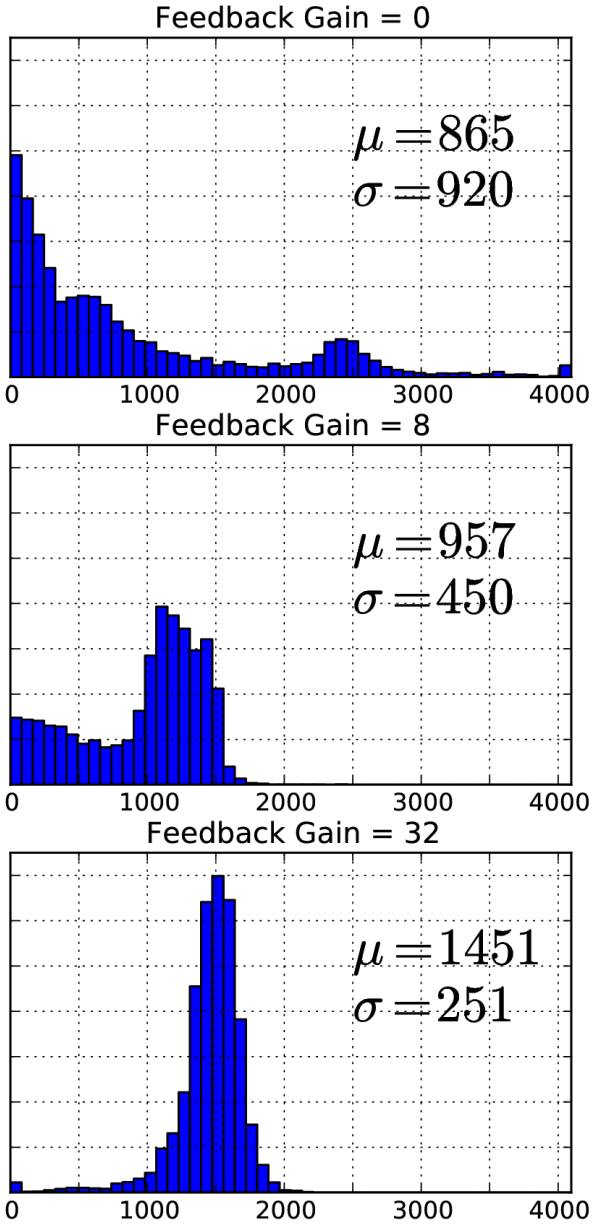

## Information flow

**FROM** Active state list (current frame)

**TO** Active state list (next frame)

# Beam Width Control

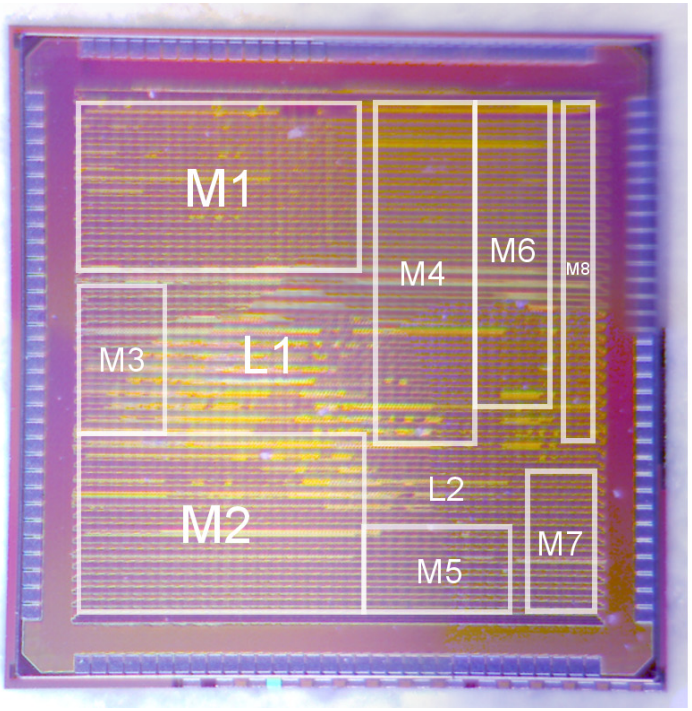



**Compress, but don't eliminate**, natural variation in active states

# Test Chip Summary

| Specification | Value |
|---|---|
| Process | TSMC 65 nm |
| Die size | 2.5 x 2.5 mm |
| Package | 128-pin LQFP |
| Logic gates | 340k (NAND2 equiv.) |
| SRAM | 2.4 Mb |
| Supply voltage | 0.7 – 1.2 V |
| Power consumption | 5 – 23 mW (core) |
| Max. clock frequency | 40 – 110 MHz |



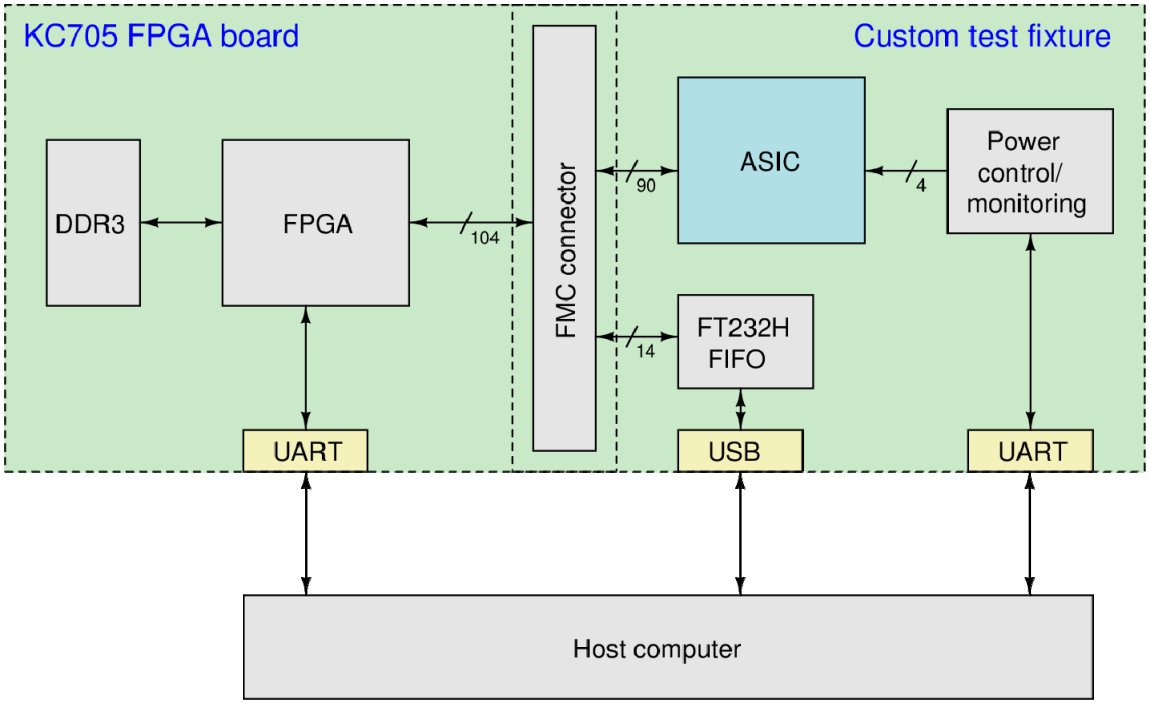

**Logic regions**

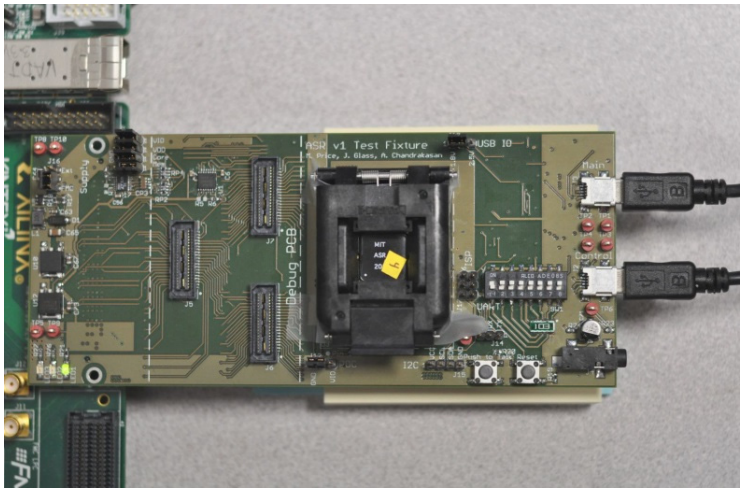
L1: Decoder
L2: Frontend

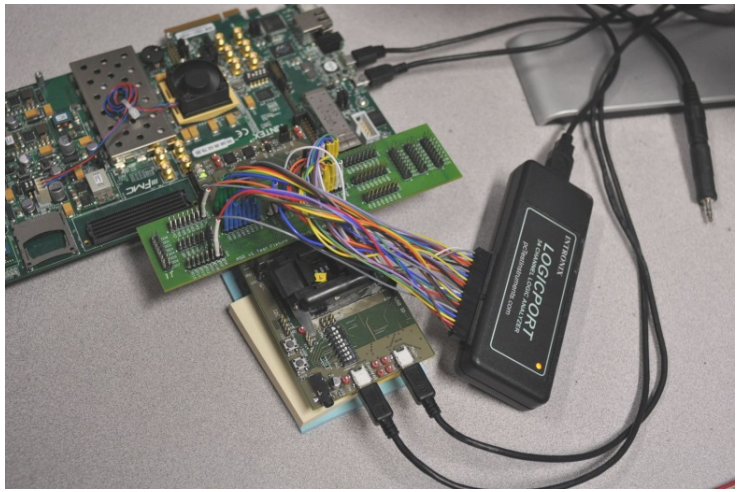
**Memories**

M1: Active state list 1
M2: Active state list 2
M3: GMM quantization tables and cache
M4: WFST cache data table
M5: Feature vector buffer
M6: WFST cache hash table
M7: Feature and audio log
M8: Frontend and FFT scratch memories

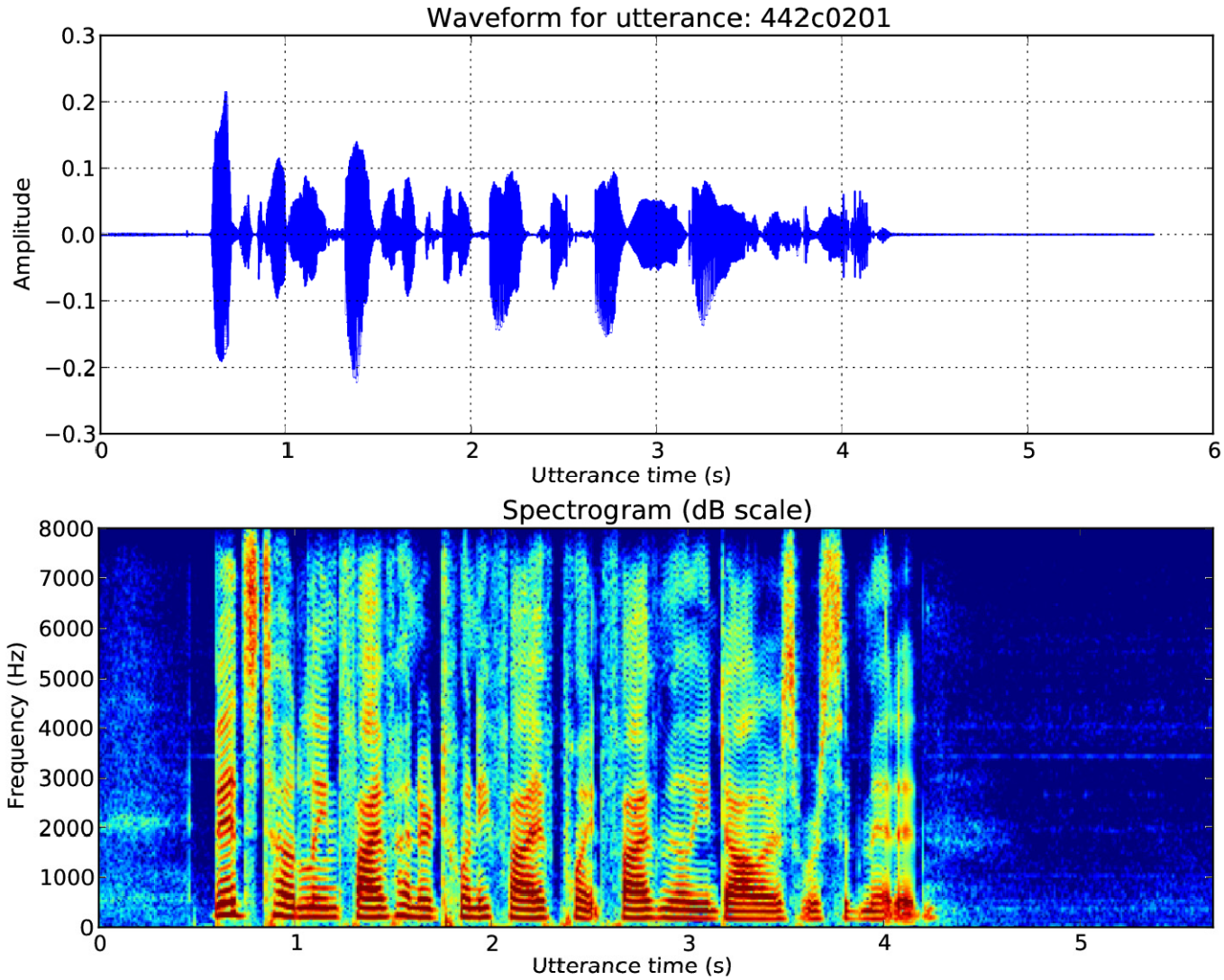
# Test Setup


KC705 FPGA board
Custom test fixture
DDR3
FPGA
104
FMC connector
90
ASIC
4
Power control/ monitoring
14
FT232H FIFO
UART
USB
UART
Host computer



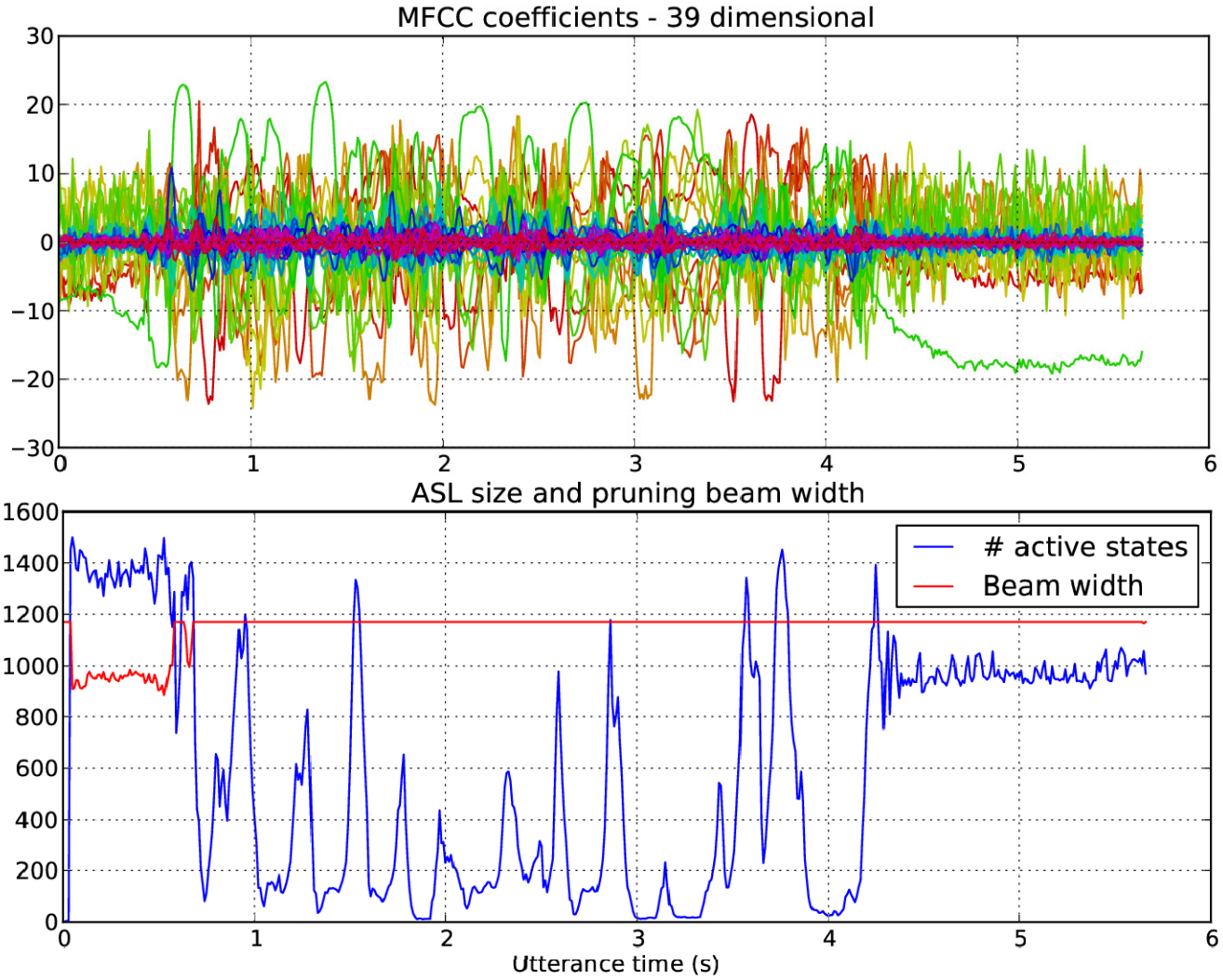


# Example: Input Audio and FFT



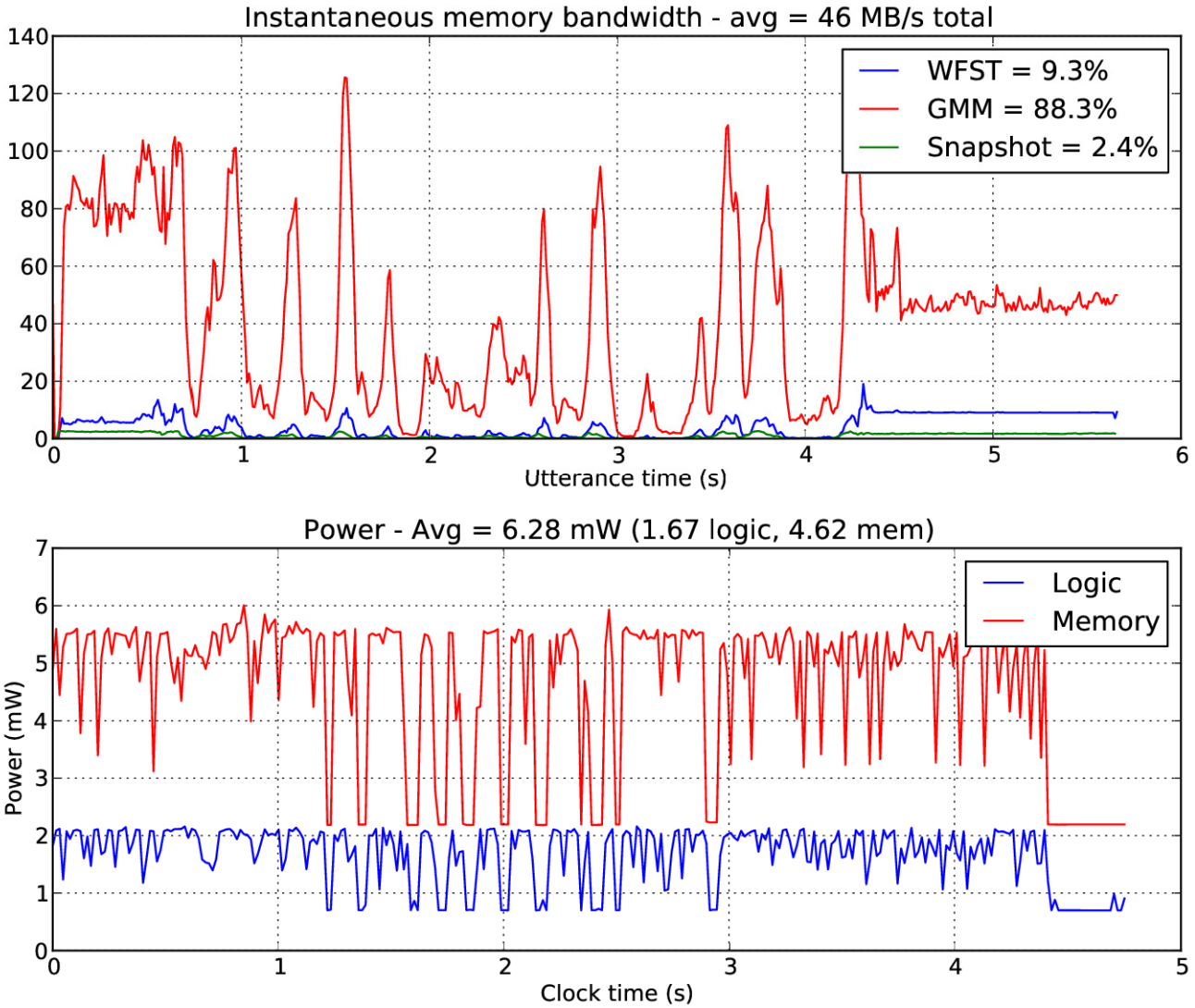

"Bids totaling five hundred twenty five point five million dollars were submitted"

# Example: Feature Vectors and Search



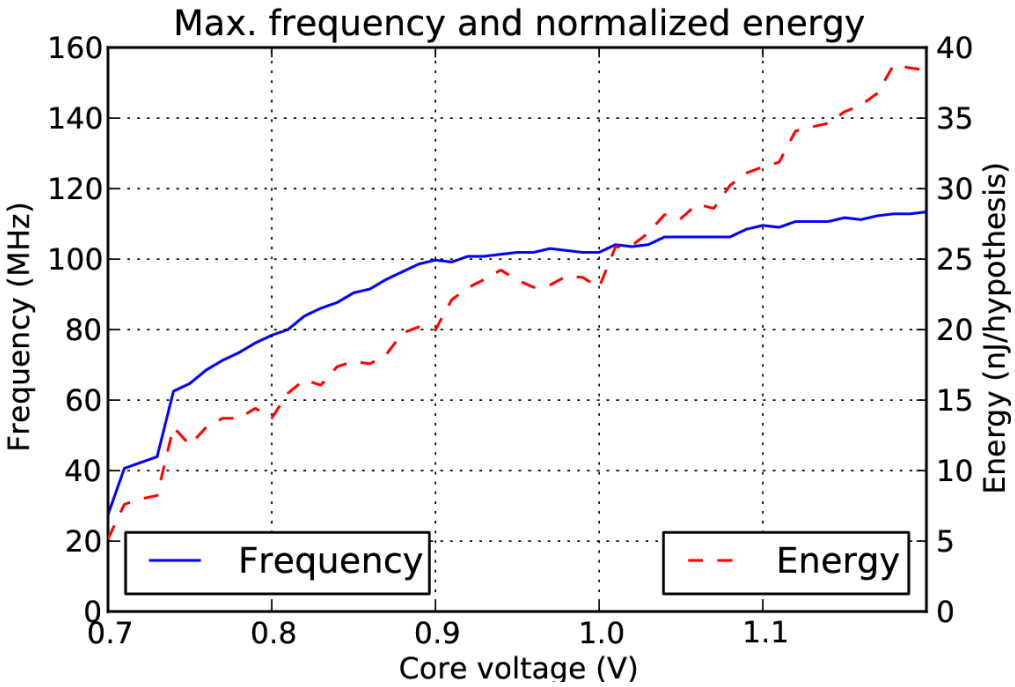

"Bids totaling five hundred twenty five point five million dollars were submitted"

# Example: Memory and Power



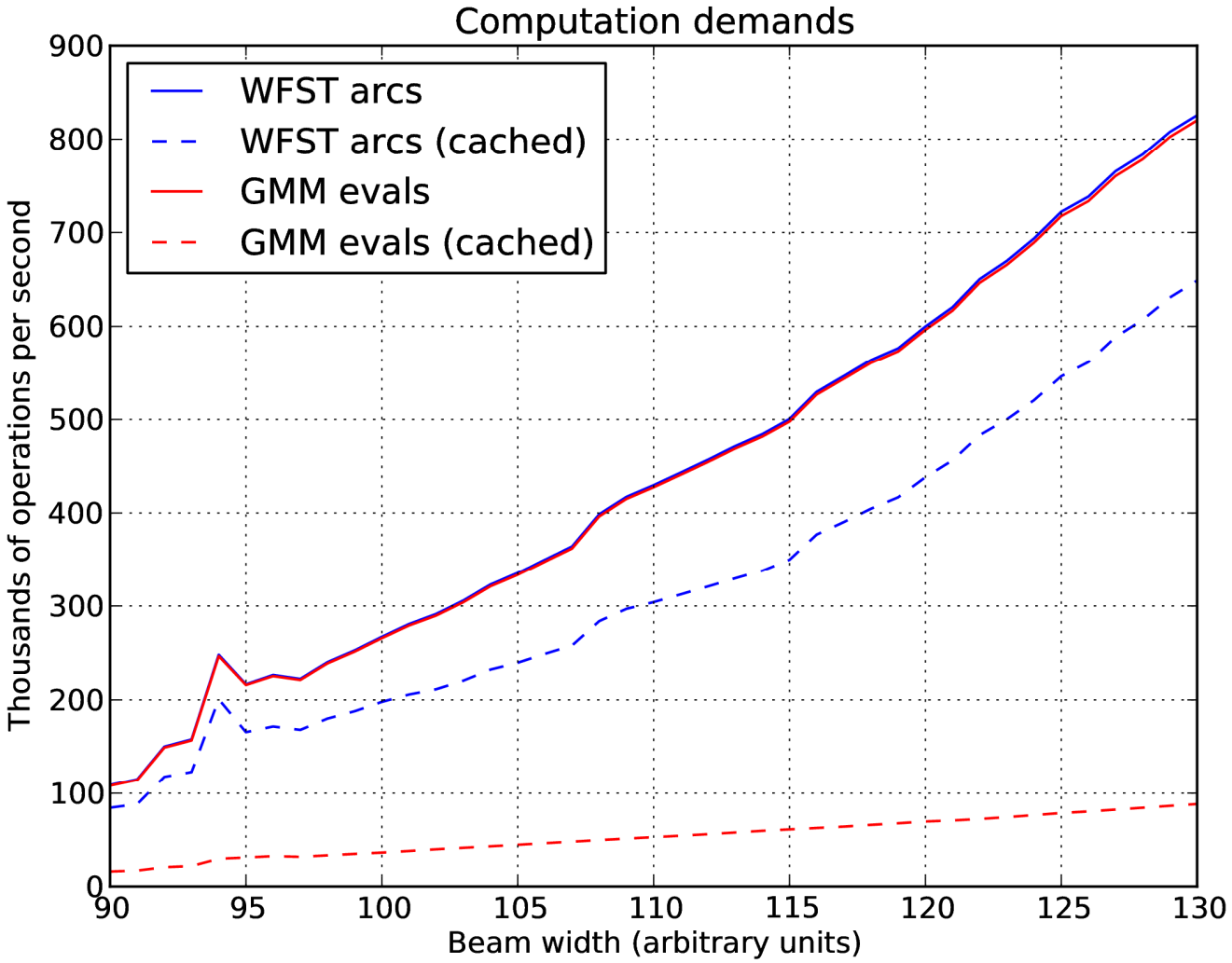

"Bids totaling five hundred twenty five point five million dollars were submitted"

# Voltage and Frequency Scaling

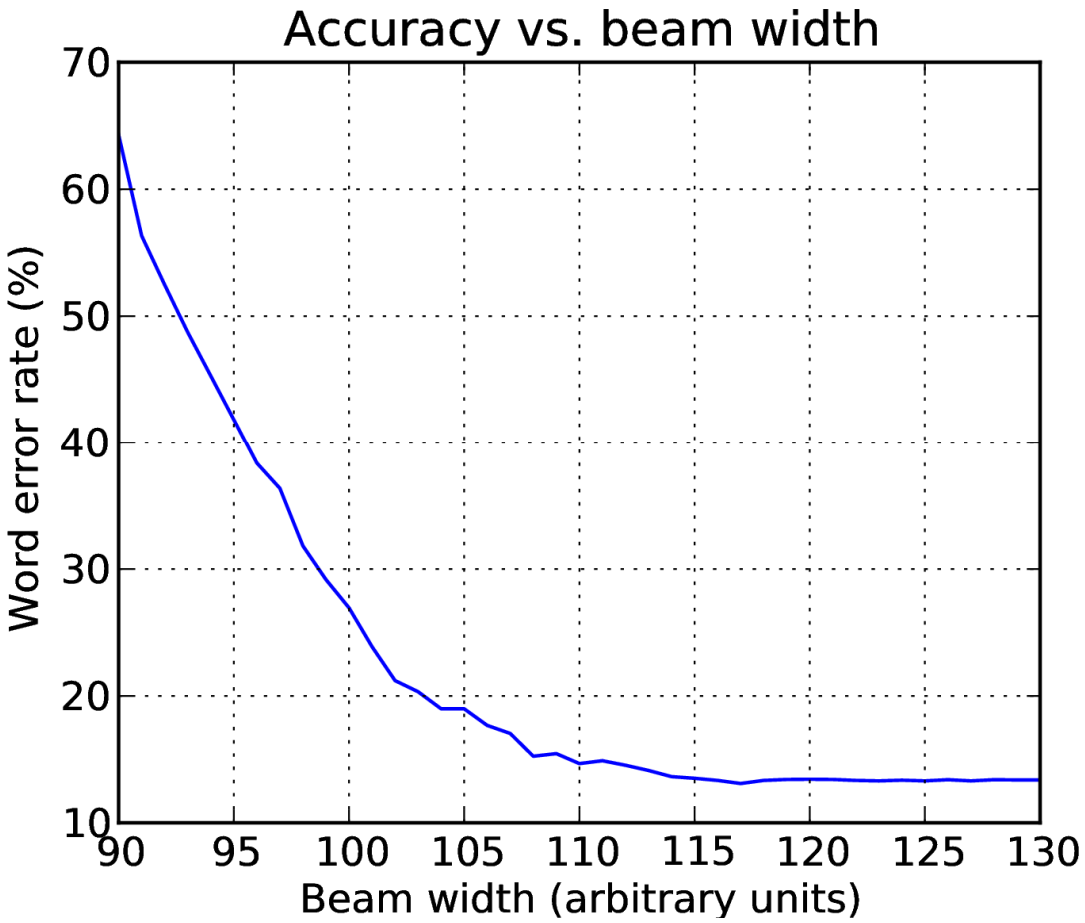



- Limited by FPGA (interface) speed above 0.9 V
- On-chip SRAM limits low-voltage scaling
- "Best" operating point:
  - 0.85 V logic, memory
  - 50 MHz
  - 13.0% WER at real-time
  - 6.0 mW average core power
- 42 µW idle leakage

# Beam Width and Performance

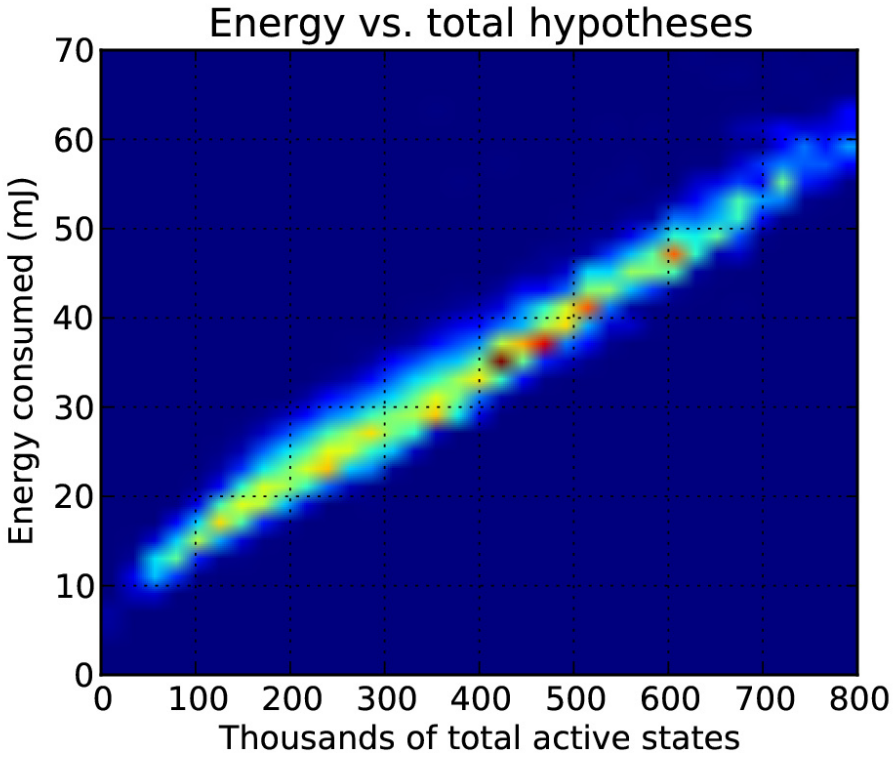


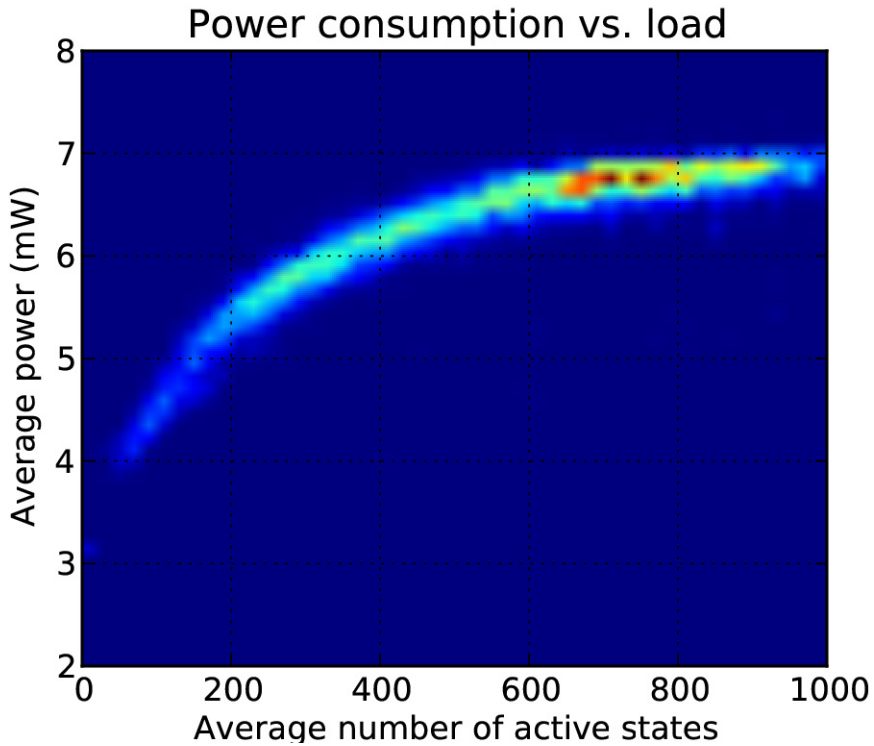
# Beam Width and Performance




**Diminishing returns** due to limitations in model accuracy

# Power Consumption Statistics




## At 50 MHz, 0.9 V

Search efficiency
16 nJ per hypothesis

# Conclusion

- End-to-end speech recognition chip
  - Includes representation and search: audio in, text out
  - Uses industry standard WFST and GMM models
  - Relies on external memory for model storage; basic control from host device (e.g. microcontroller)

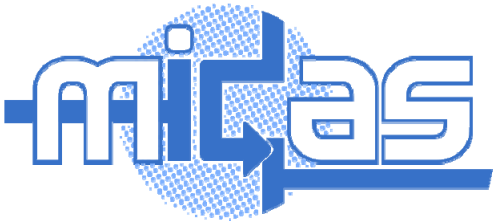
- Architectural enhancements for low-power system
  - Front-end signal processing transformations (reduced area and power)
  - WFST and GMM parameter compression and caching (reduced memory bandwidth)
  - Beam width control via feedback (reduced on-chip memory size)

**Acknowledgements:**
Quanta Computer, Inc.
Irwin and Joan Jacobs fellowship
TSMC University Shuttle Program

# A 210mV 5MHz Variation-Resilient Near-Threshold JPEG Encoder in 40nm CMOS

*N. Reynders and W. Dehaene*

KU LEUVEN



KU Leuven, ESAT-MICAS, Belgium

# Outline

- Introduction
- Gate topology
- Design of JPEG encoder
- Measurement results
- Conclusion

# Introduction

- Near-threshold DSP
  - ➡ large energy savings
- Challenges:
  - Large delay ➡ moderate performance
  - High variability ➡ variation-resiliency needed

# Introduction

- Near-threshold DSP
  - ➡ large energy savings
- Challenges:
  - Large delay ➡ moderate performance
    - Higher performance in advanced nanometer CMOS
  - High variability ➡ variation-resiliency needed
    - Worse in advanced nanometer CMOS

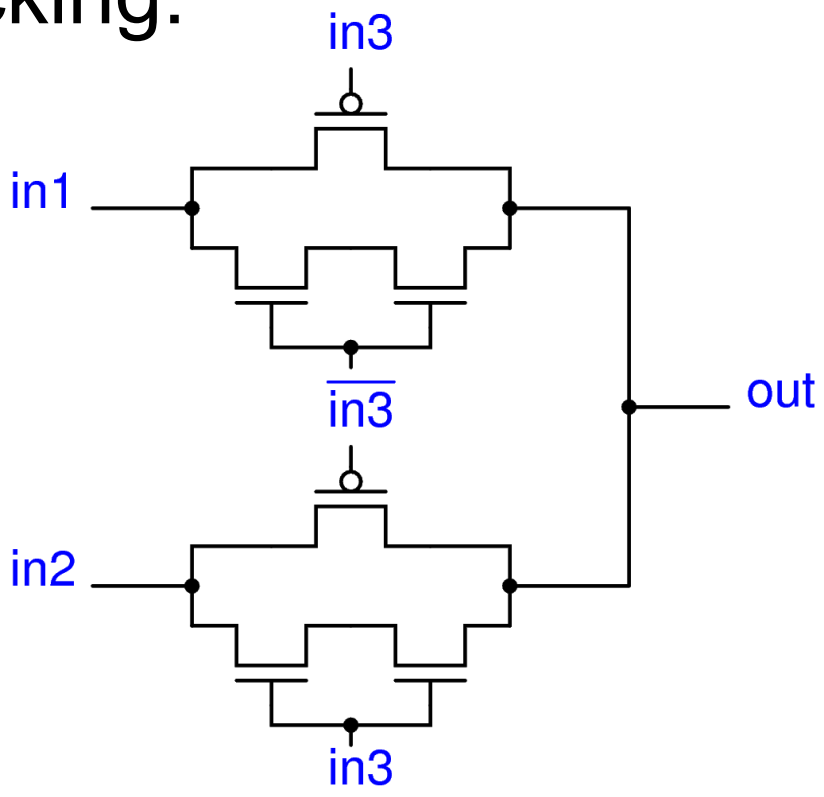
➡ Attractive to go to 40nm CMOS

# JPEG Encoder

- Representative DSP block
- Target:
  - Very low energy consumption
  - Clock frequencies in MHz-range
  - High variation-resilience

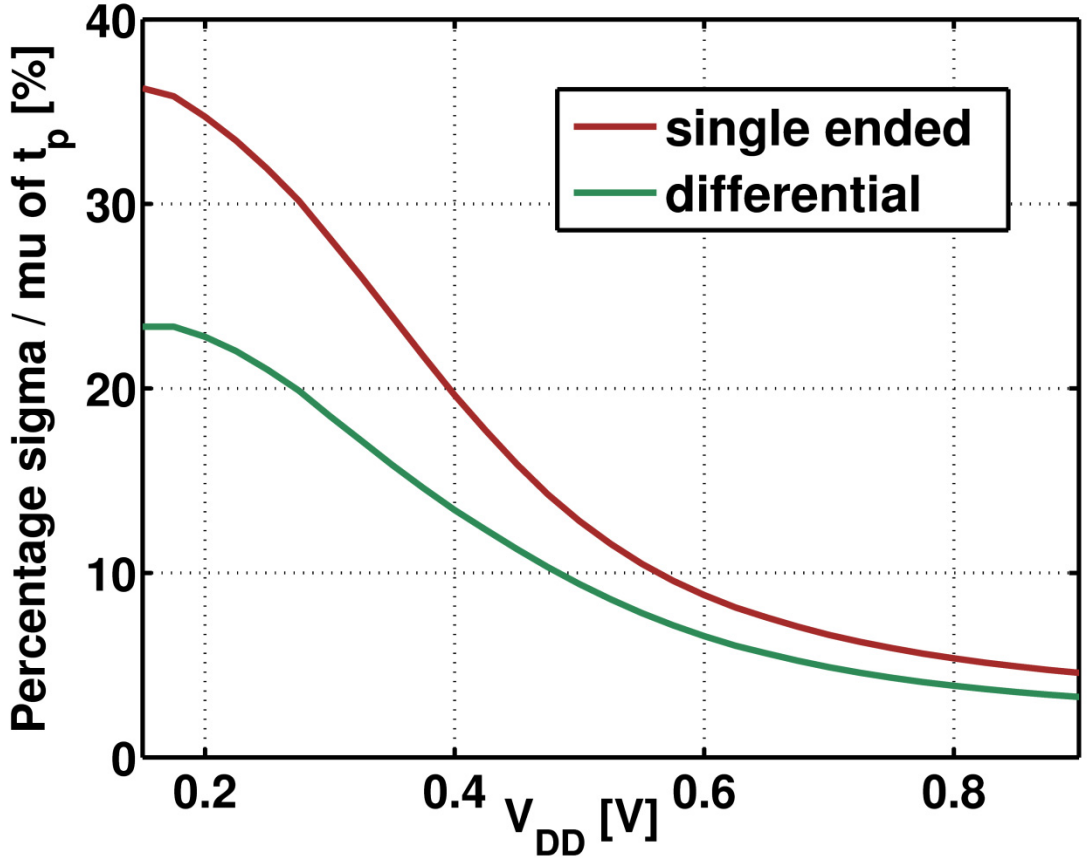
# Gate topology

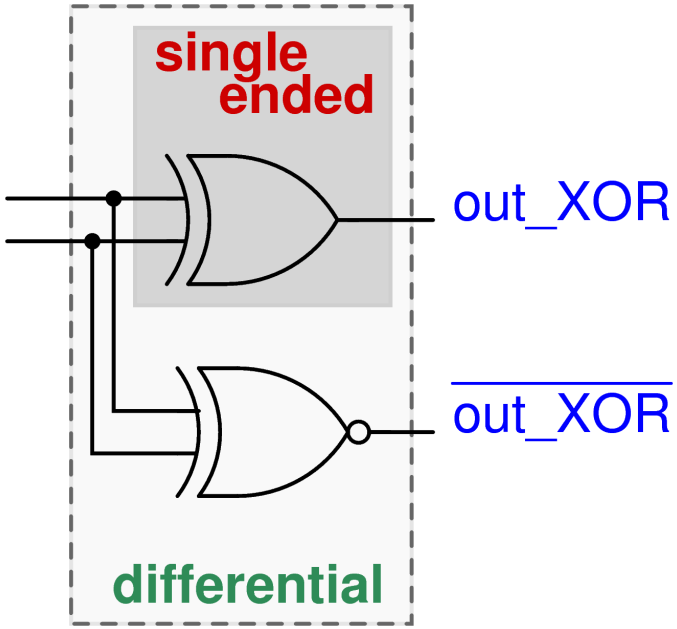
- Critical in ultra-low-voltage design
- Transmission Gate (TG) logic extended with nMOS stacking:
  - No direct leakage paths
  - Variation-resilient:
    - nMOS + pMOS in each conducting path
    - Differential implementation
  - Generic building block



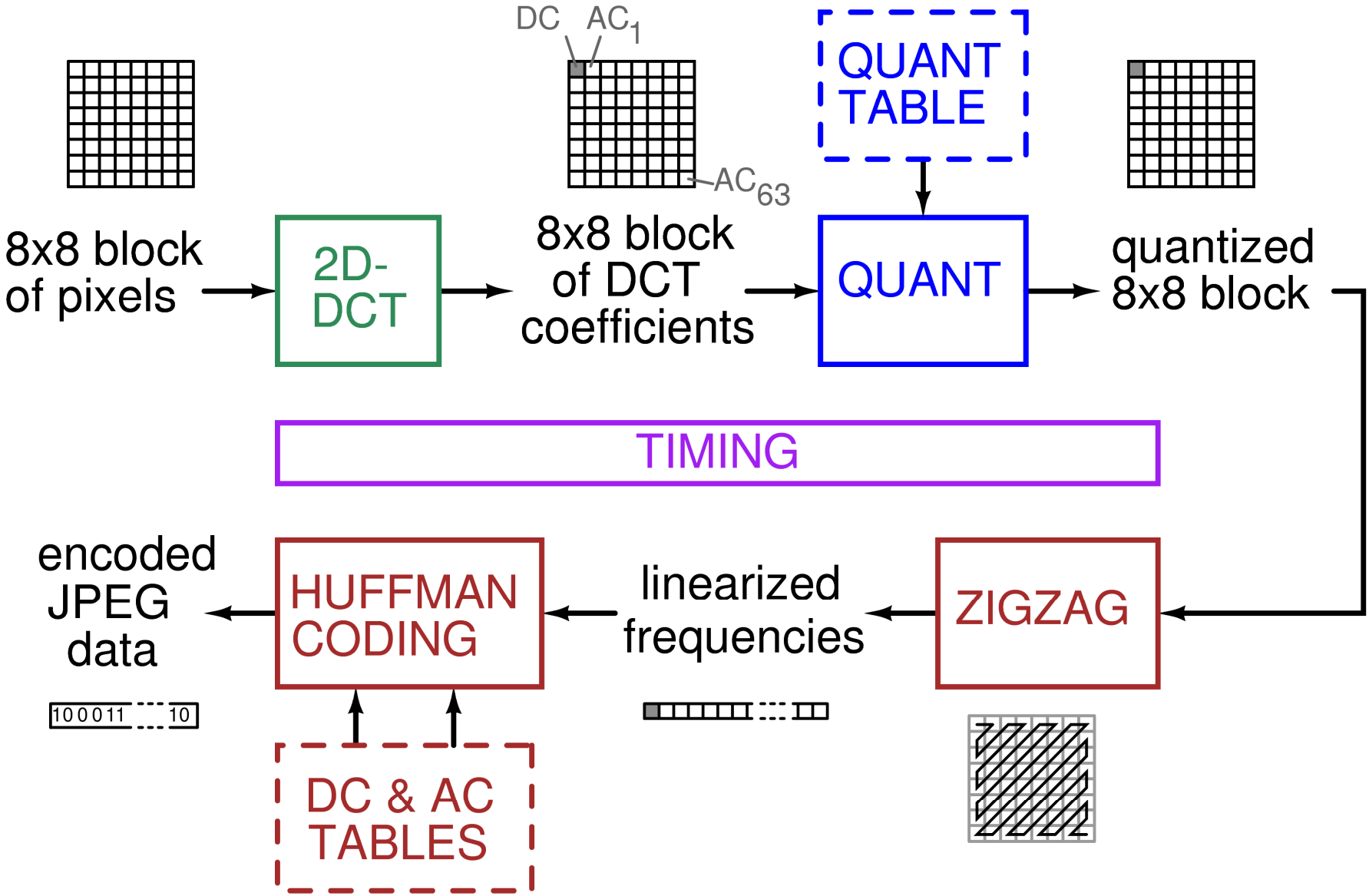
# Gate topology

- Differential implementation:
  - Better variation-resilience:
    - Variation of propagation delay lower

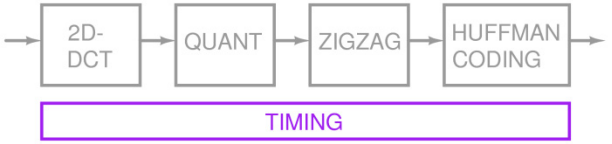


# JPEG encoder: overview

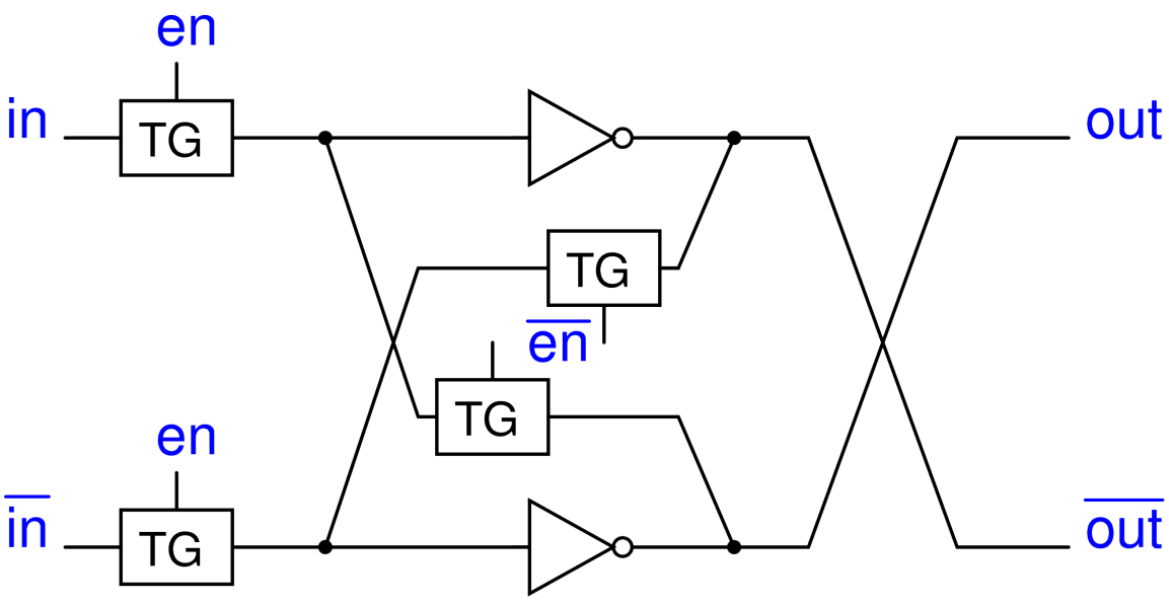


# Timing

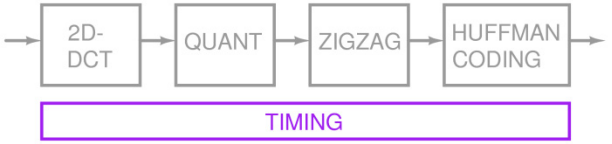



- Latch-based pipelined architecture:
  - 1 Clock domain & Non-overlapping clocks
  - Clock tree
  - Allows time borrowing
  - Differential latch

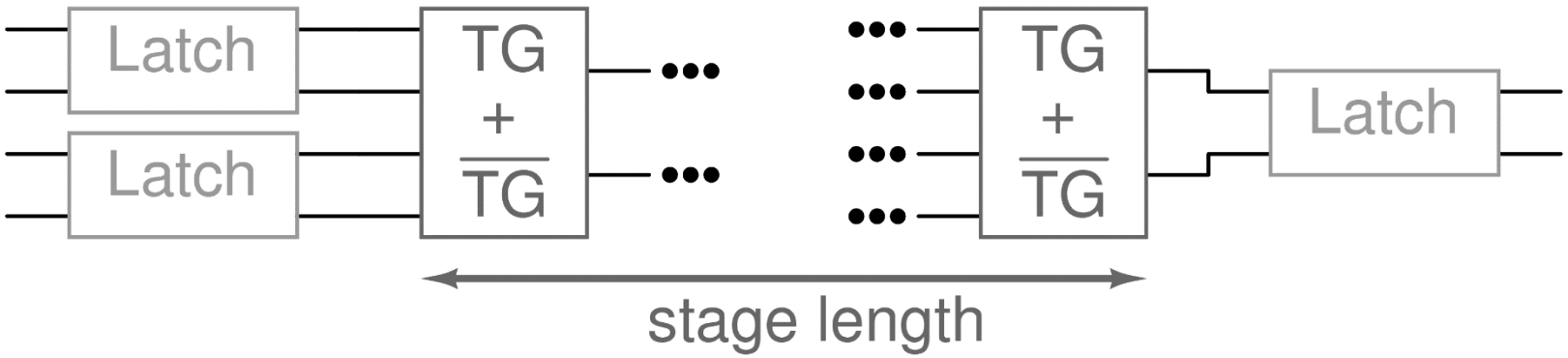
➡ variation-resilient



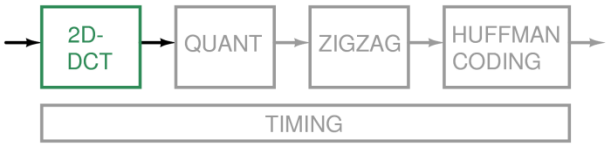
# Pipeline stage

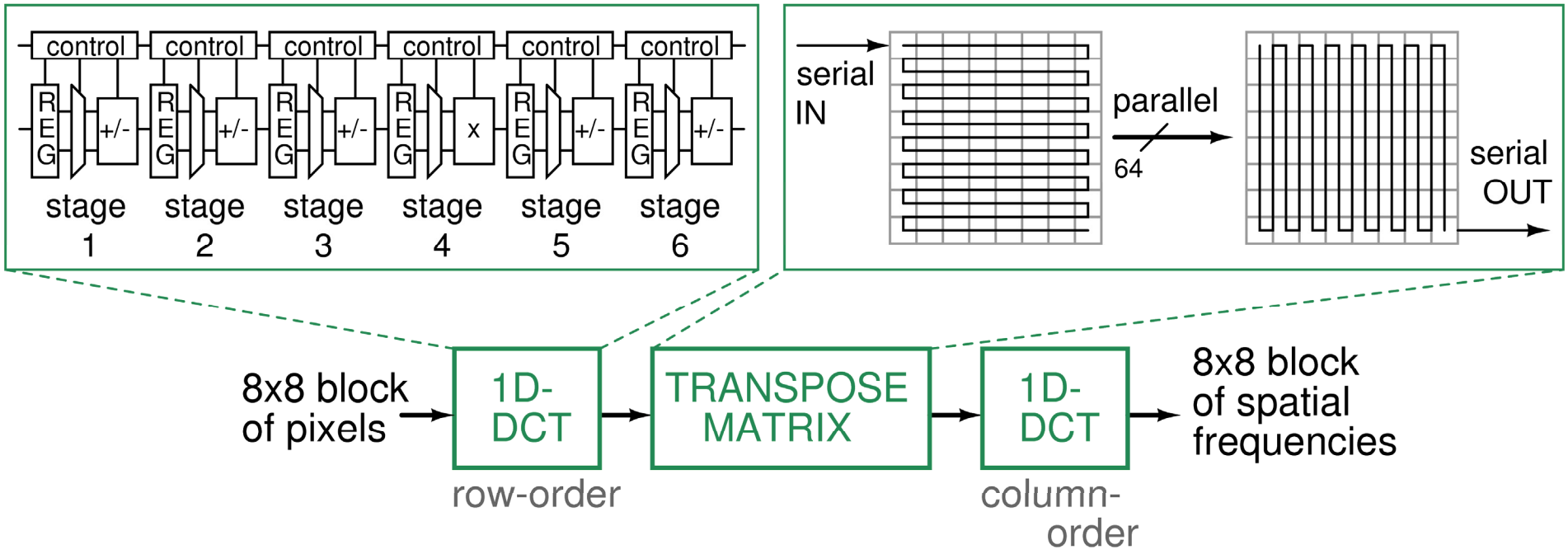



- Stage length trade-off:
  - Average timing variations: cascading logic gates
  - Guarantee robustness: signal loss
- Max. stage length = 3 ➡ Deep pipelining



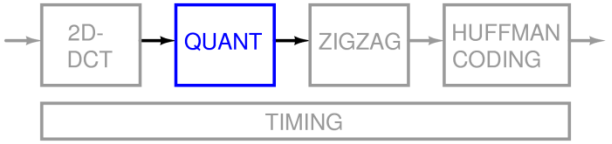
# 2D-DCT

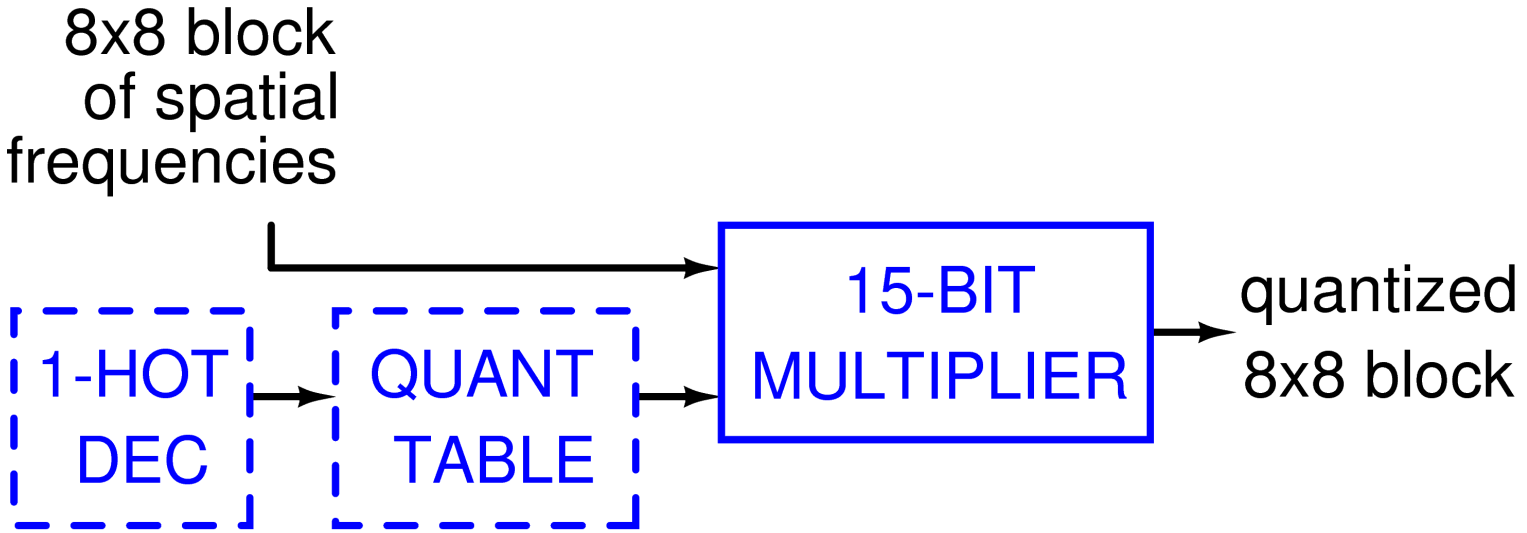



- Pipelined 1D-DCT algorithm, based on [Kovac]
  - Five 15b adder/subtractors
  - One 15b multiplier
- Transpose matrix: two 8x8 blocks of registers

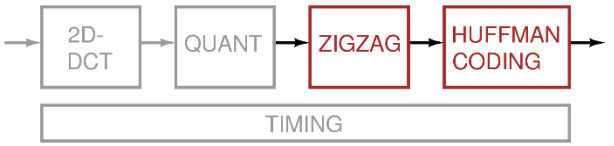
# Quantization

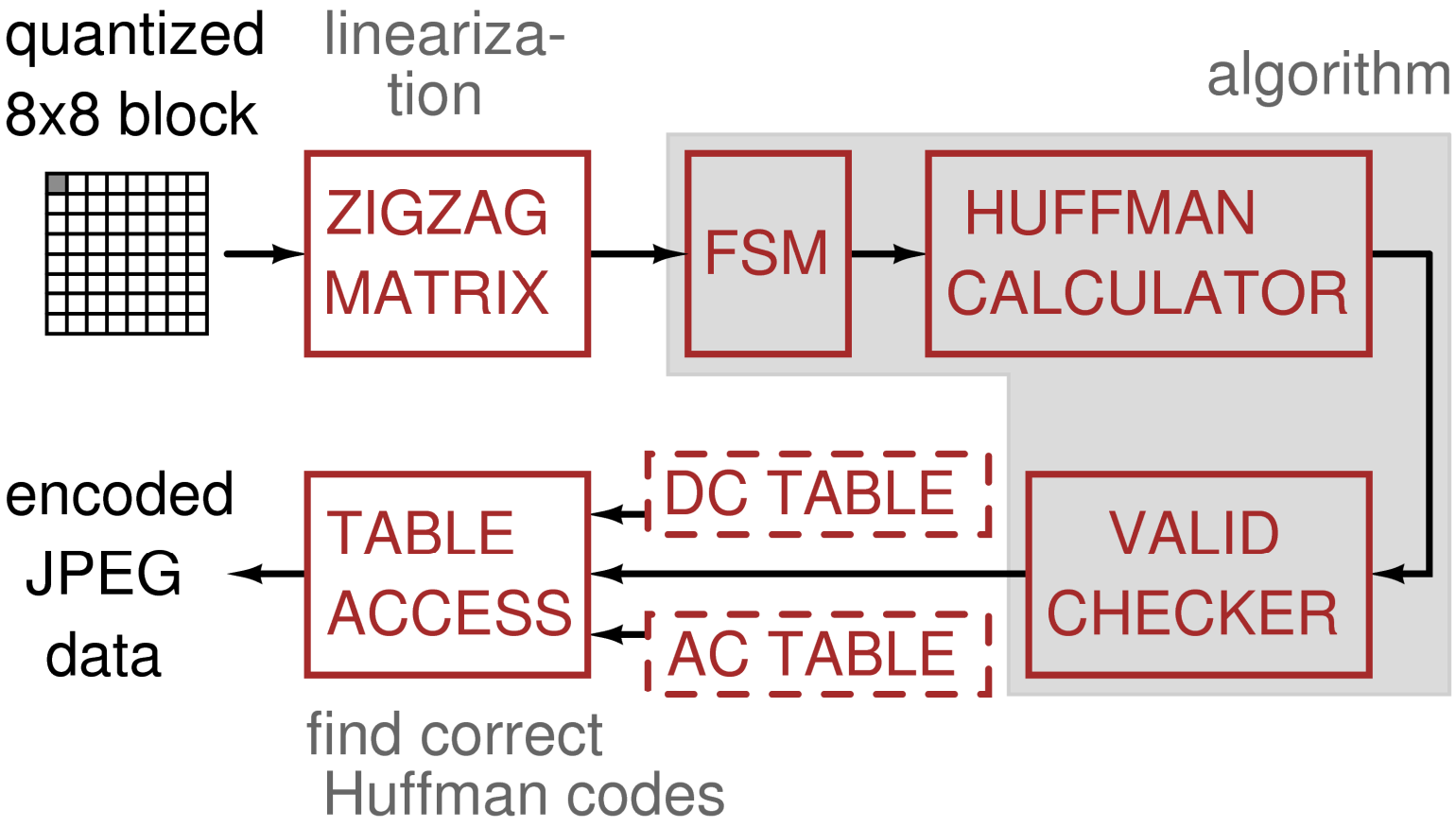



- Quantization coefficients: scaled for DCT
- Stored in table
- Accessed by one-hot decoder

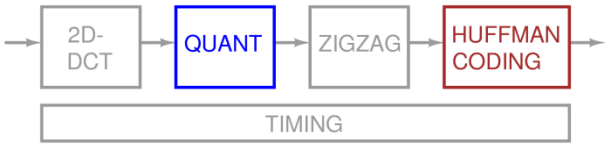
# Huffman encoder



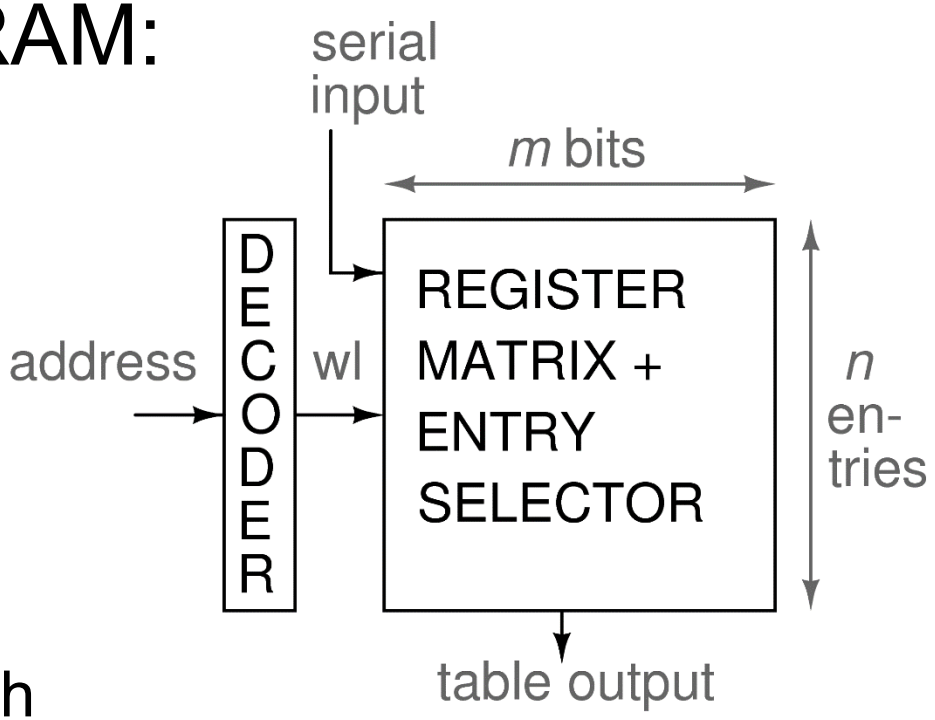




- Zigzag matrix similar to transpose matrix
- Regular decoder for DC/AC Huffman tables

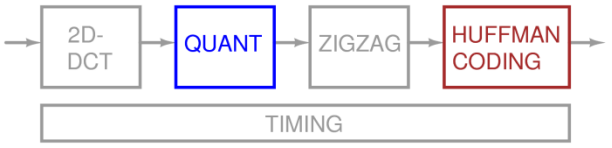
# Tables



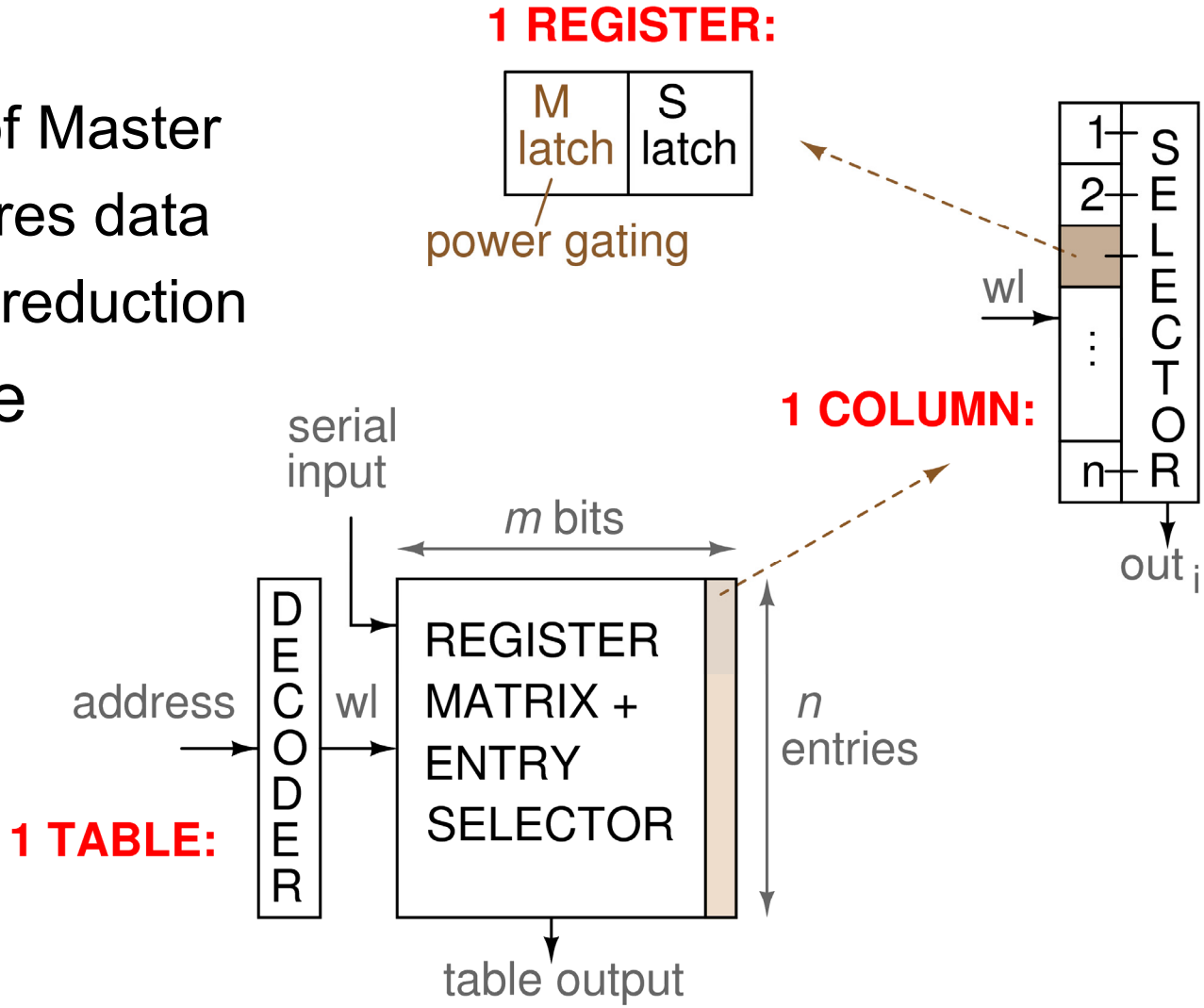

- Register tables, not SRAM:
  - Dominated by standby leakage, not by dynamic energy
  - For required # of bits:
    - Peripheral area too high
    - Energy overhead too high
  - Near-threshold SRAM speed too low
  - Efficient SRAM requires ratioed logic:
    - Undesirable due to variation sensitivity



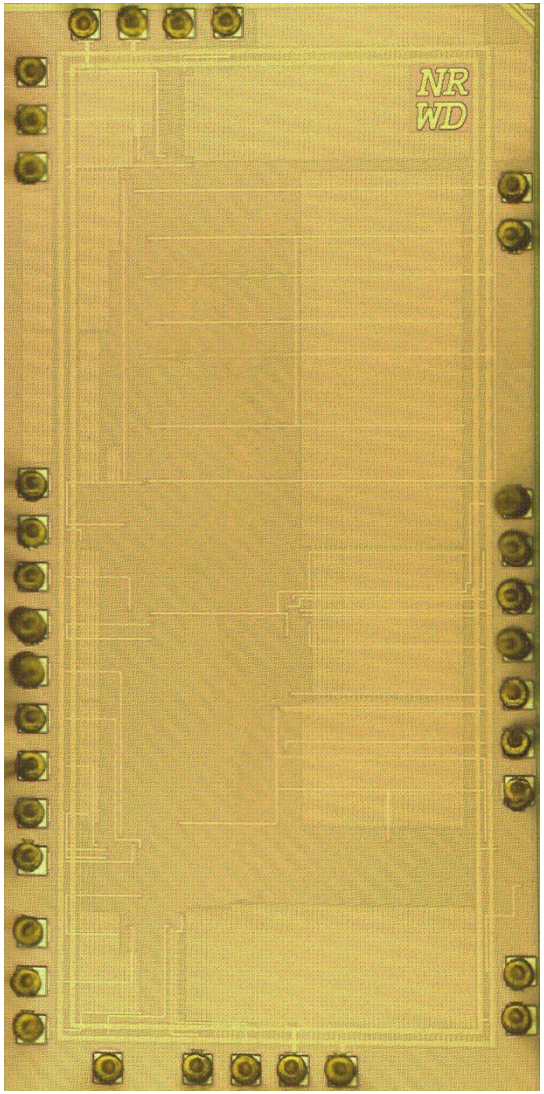
# Tables




- Startup: Data serially clocked in
- After startup:
  - Power gating of Master
  - Slave latch stores data
  - Large leakage reduction
- Decoder & table entry selector:
  - Max 3 TG + latches



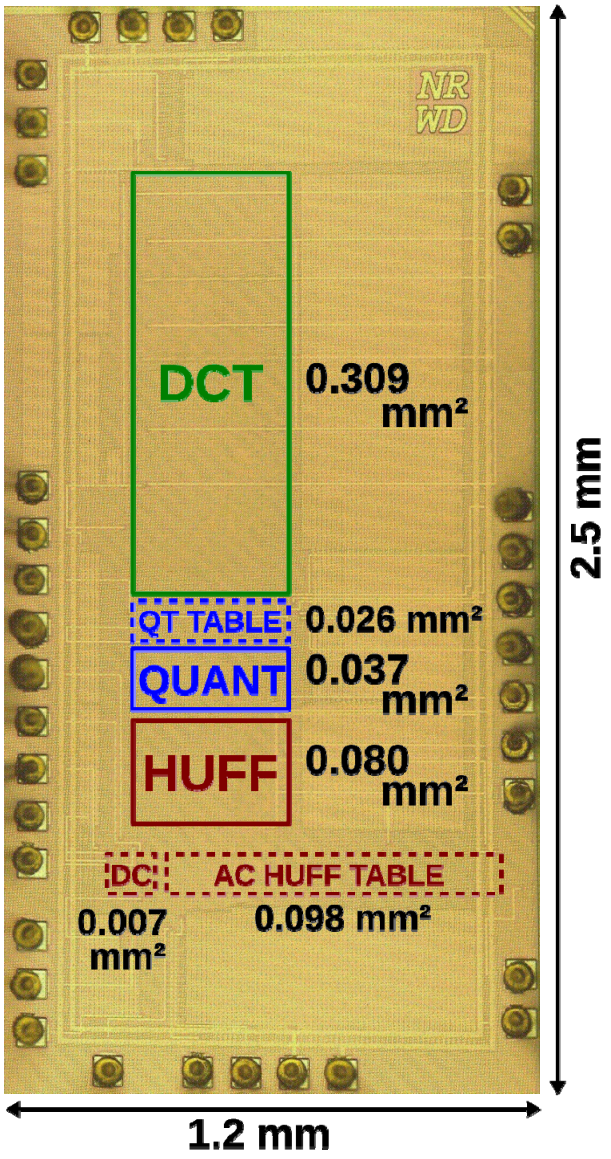
# Chip




- 40nm CMOS
- Fully functional

Encoded JPEG picture (through the chip) of the chip

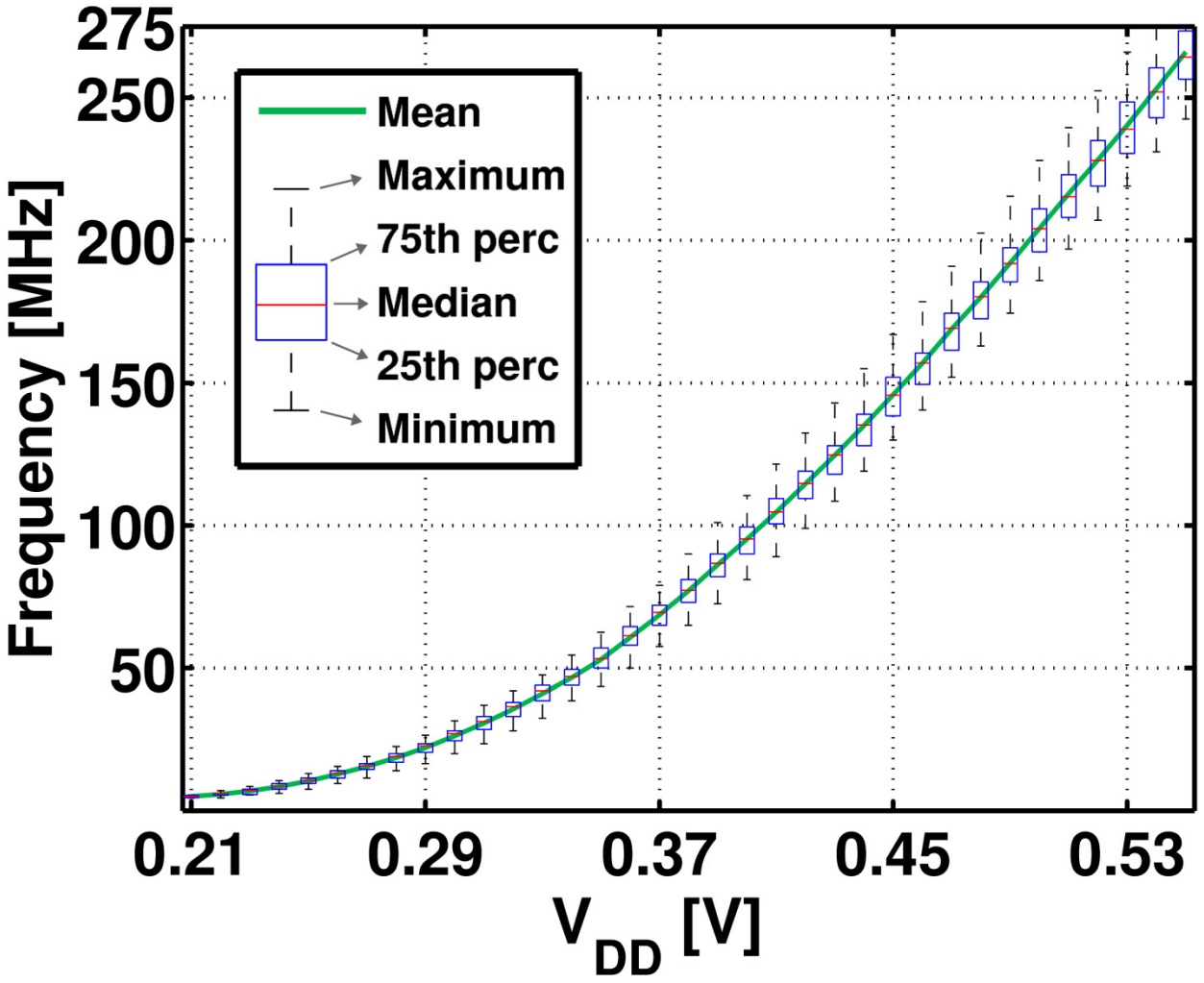
# Chip




| | Pipeline Depth |
|---|---|
| 2D-DCT | 458 |
| Quantization | 22 |
| Zigzag | 130 |
| Huffman coding | 36 |
| **Total JPEG** | 646 |

- Total active area = 0.557mm²
- Dense layout: Datapath Generator
  - Exc. tables: manual

# Operating frequency



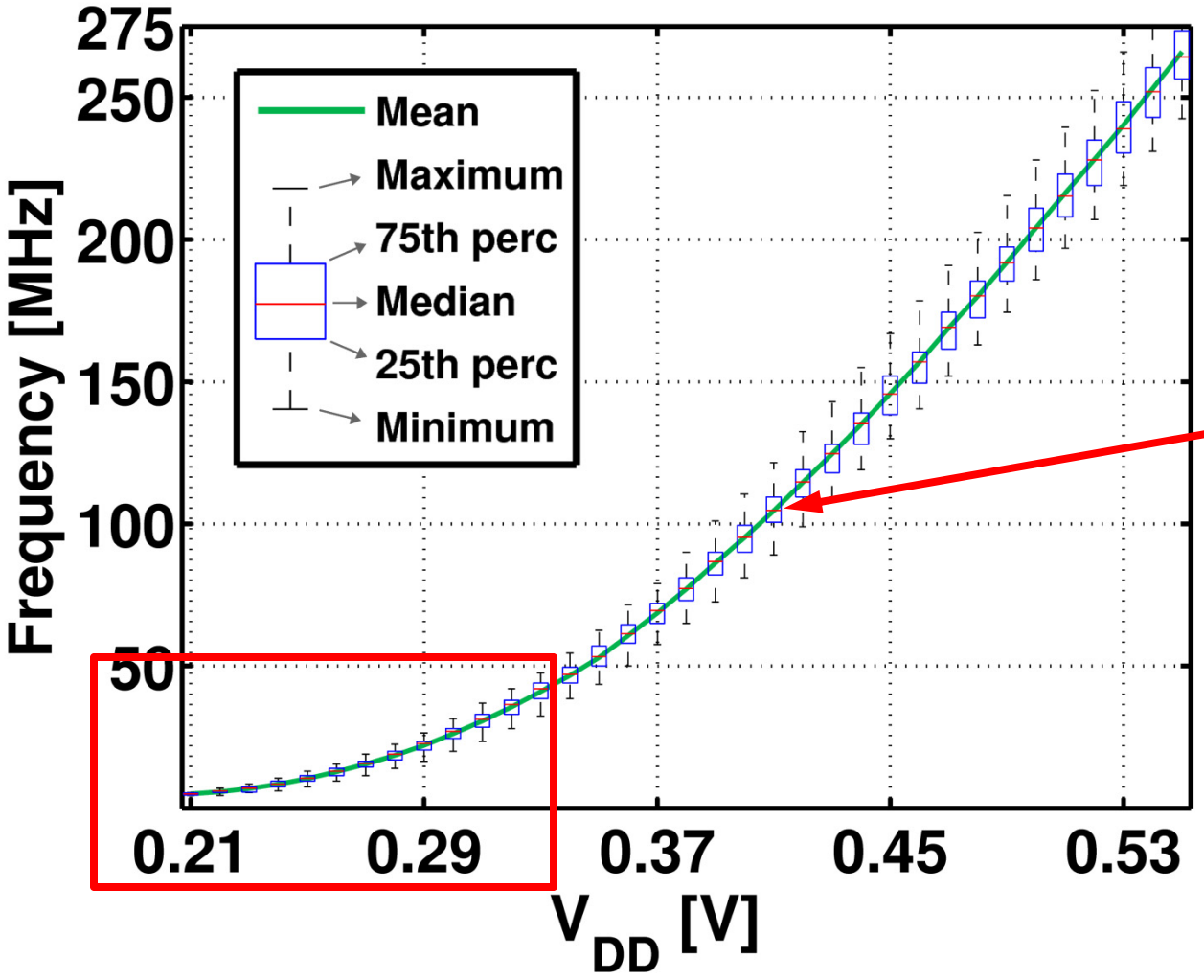

Functional down to $V_{DD}$=210mV

➡ f = 5MHz

26 dies:

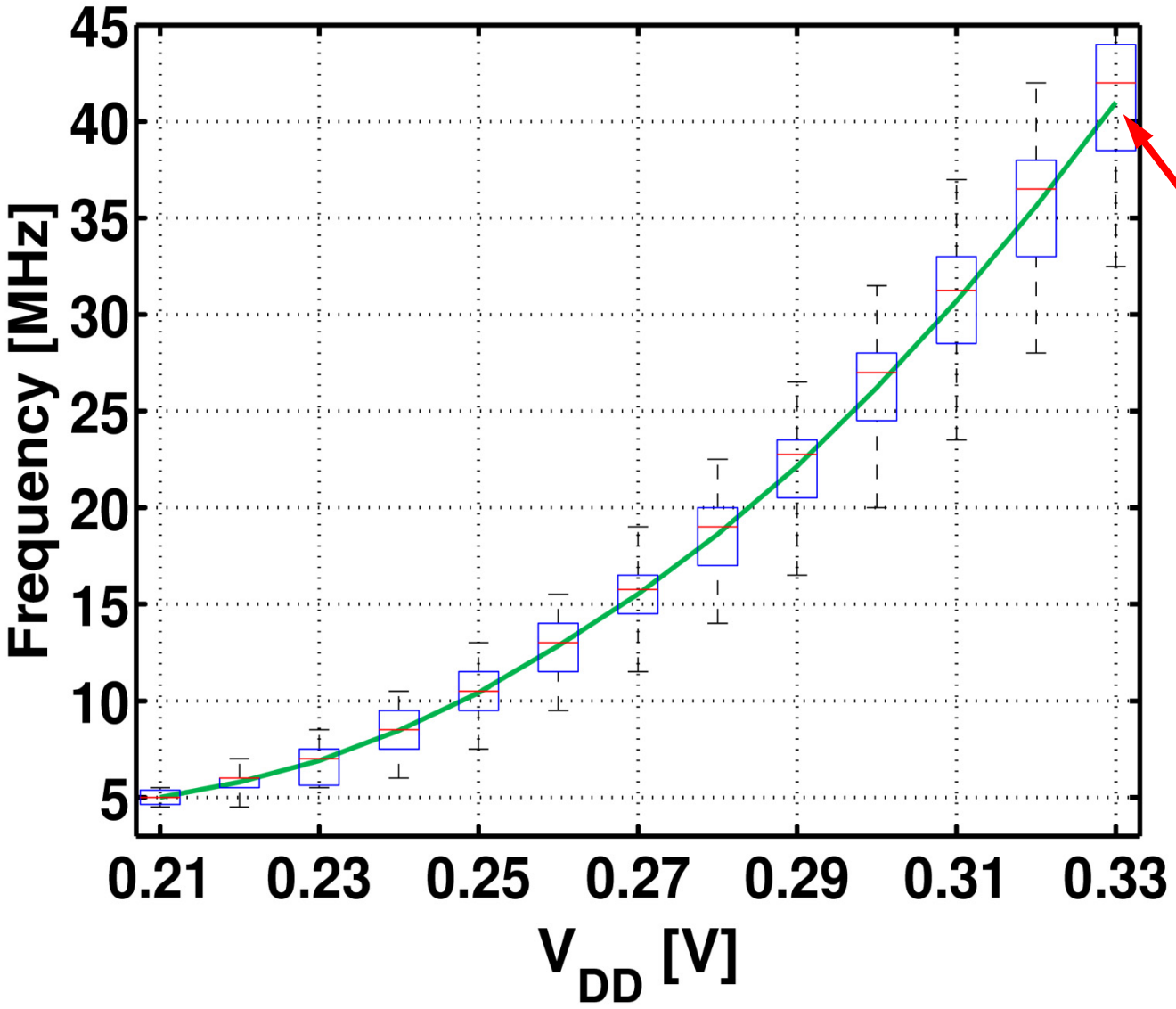
σ/μ = 8.6%

# Operating frequency



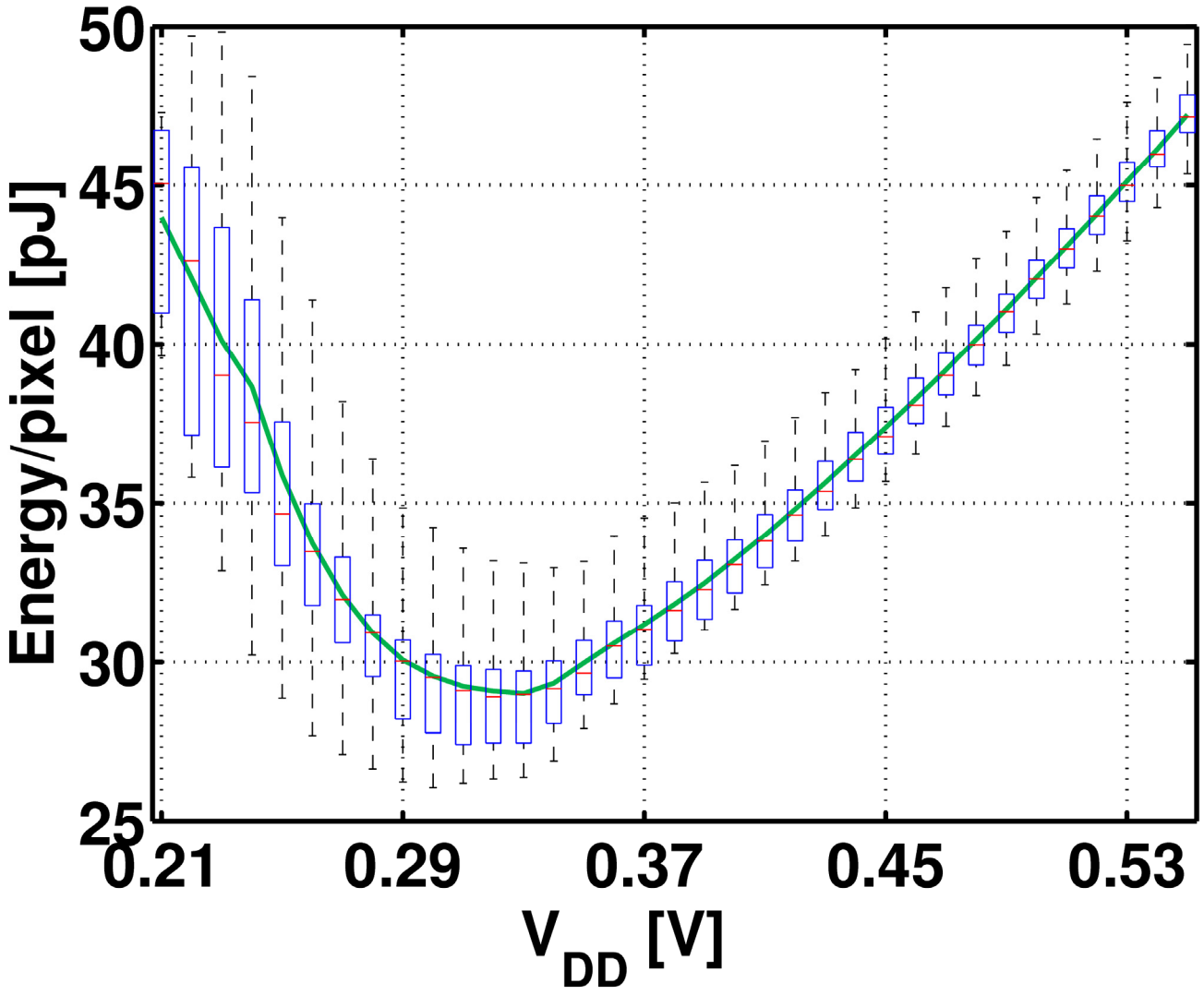

100MHz @
$V_{DD}$=410mV

# Operating frequency




41MHz @
$V_{DD}$=330mV

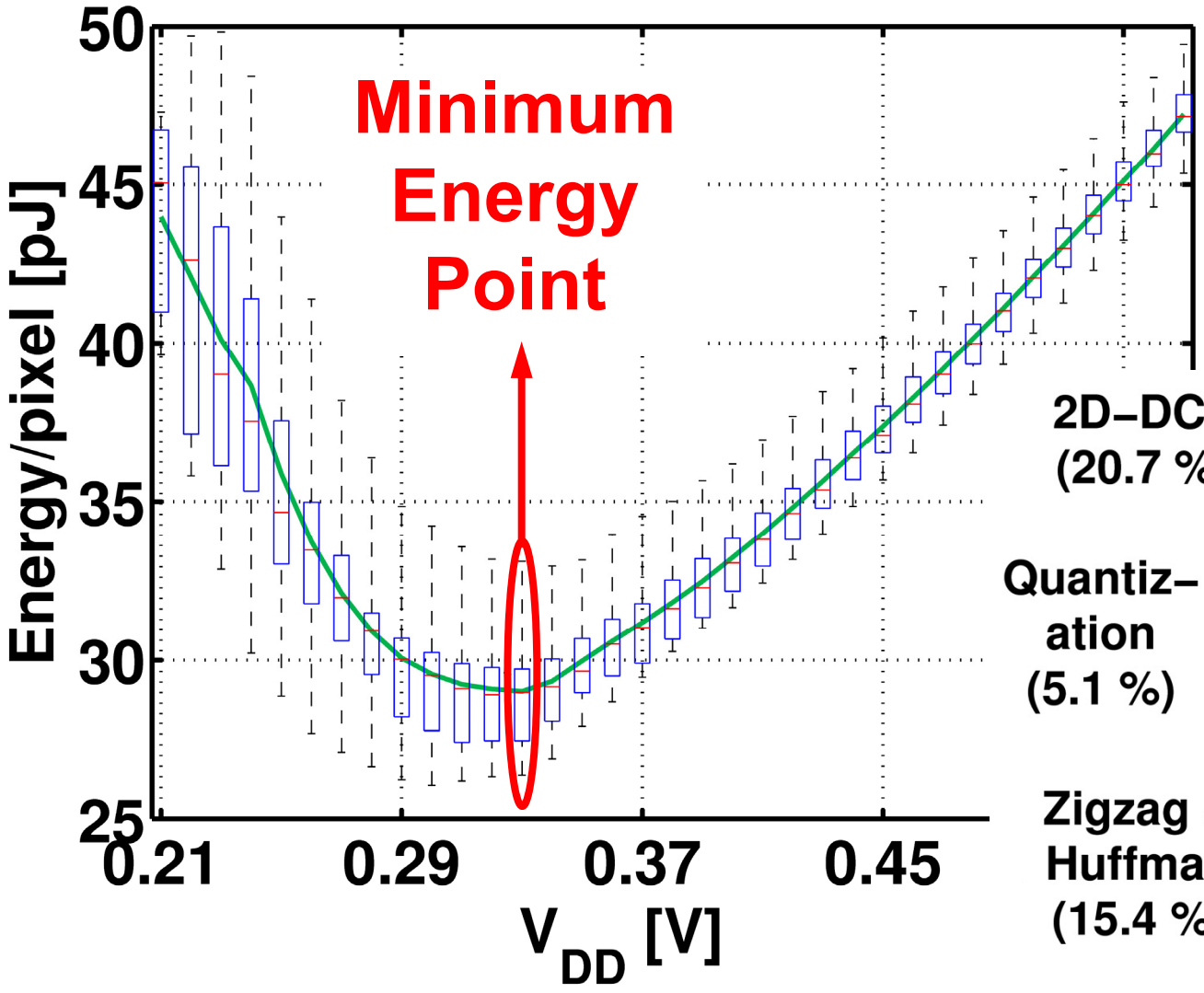
# Energy consumption



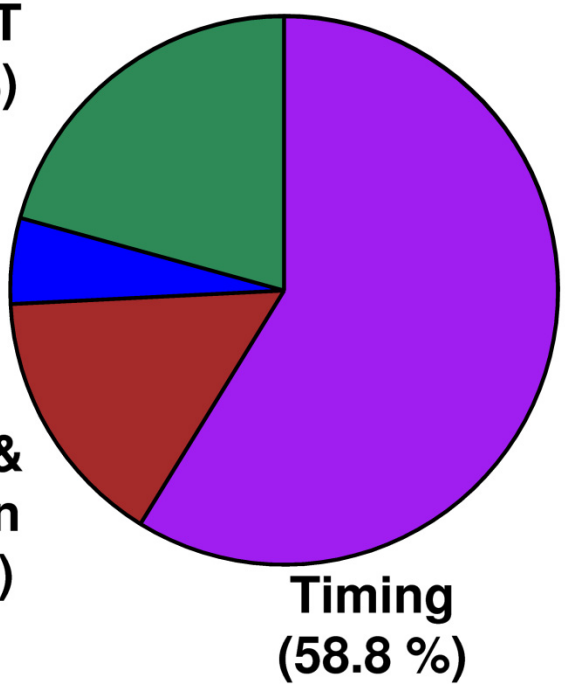

26 dies:

σ/μ = 5.4%

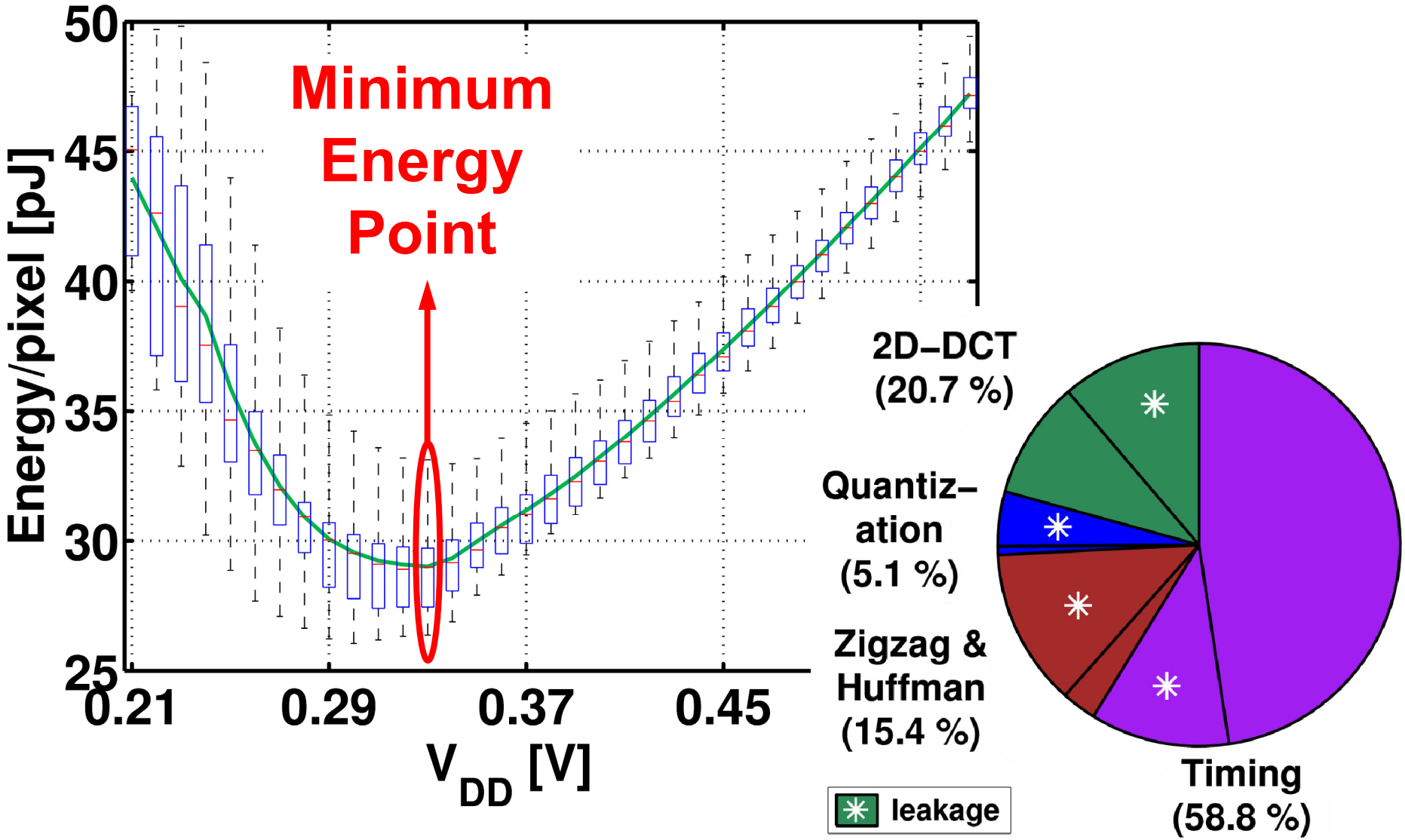
# Energy division at MEP



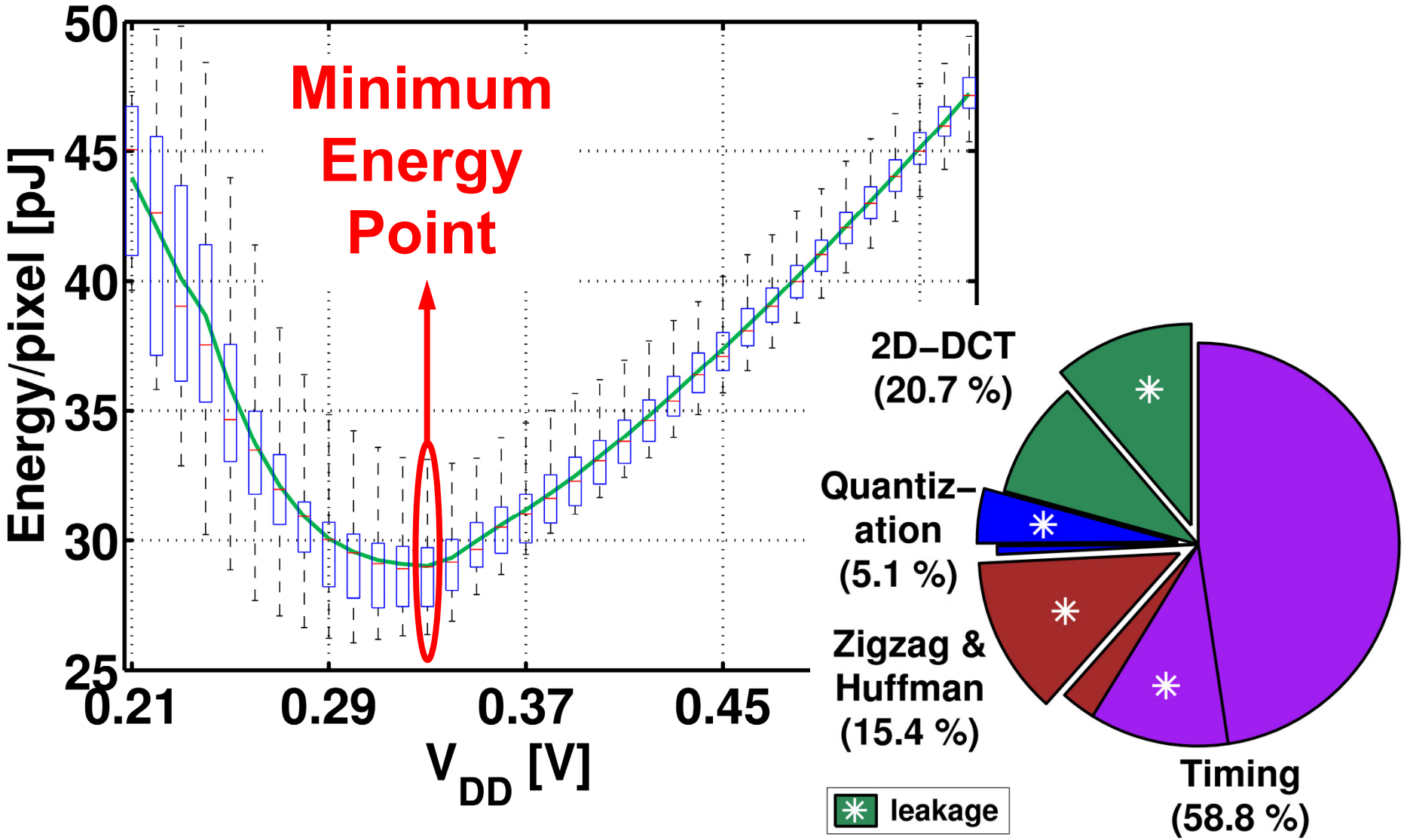

29.01pJ @
$V_{DD}$=330mV



# Leakage contribution



# Leakage contribution

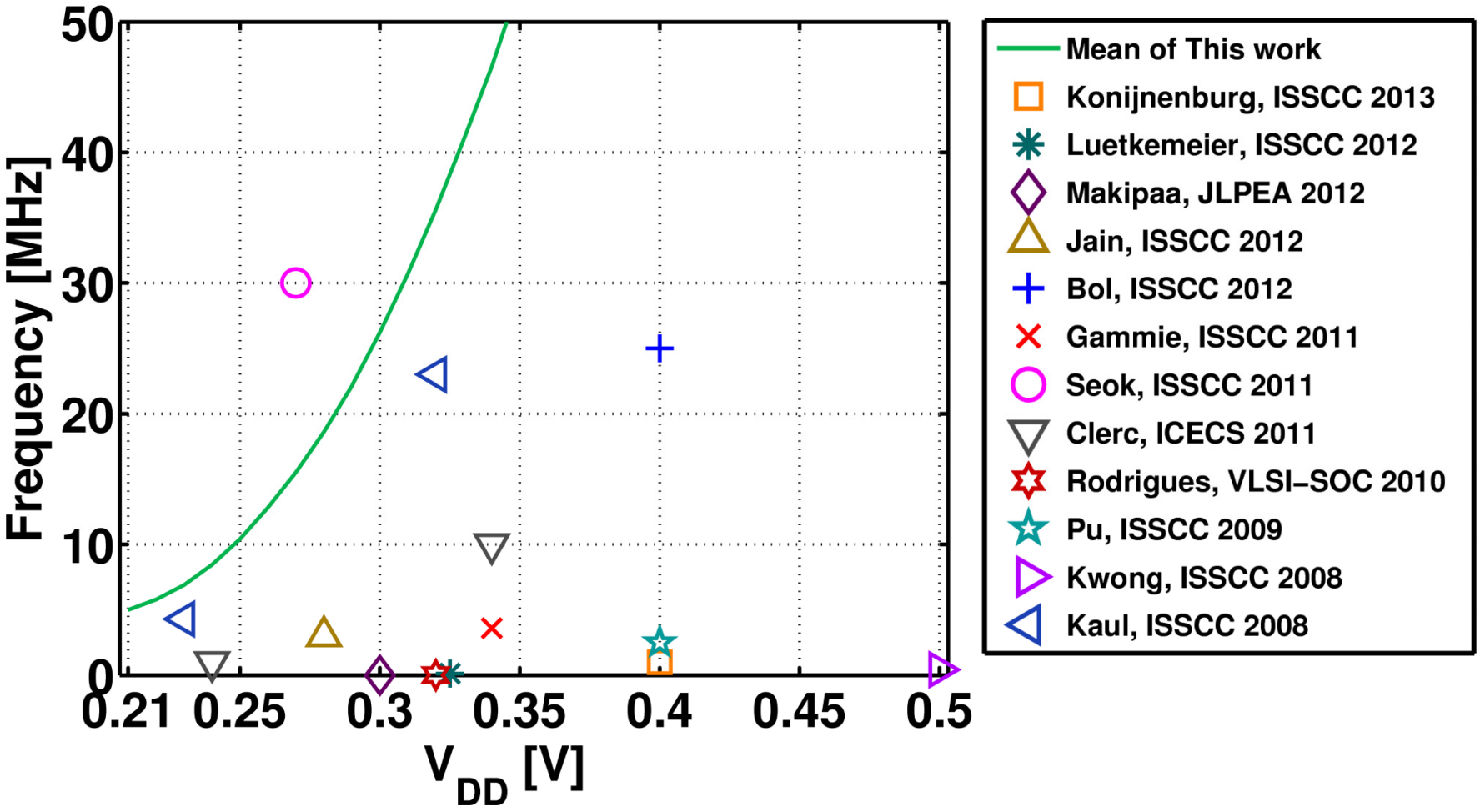


# State-of-the-art comparison

| | | This work | [6] |
|---|---|---|---|
| **CMOS Technology** | | 40nm | 65nm |
| **Active area** | [$mm^2$] | 0.557 | 1.960 |
| **Minimal voltage** | [V] | 0.210 | 0.400 (engine) / 0.600 (Huff) |
| **Voltage @ MEP** | [V] | 0.330 | 0.400 / 0.600 |
| **Frequency @ MEP** | [MHz] | 41.0 | 2.5 / 10 |
| **Energy/cycle @ MEP** | [pJ] | 0.045 | 3 (for 4 engines) + 1.8 |
| **Energy/8x8 block @ MEP** | [pJ] | 1857 | - |
| **EDP @ MEP** | [pJ.$\mu$s] | 45.29 | - |

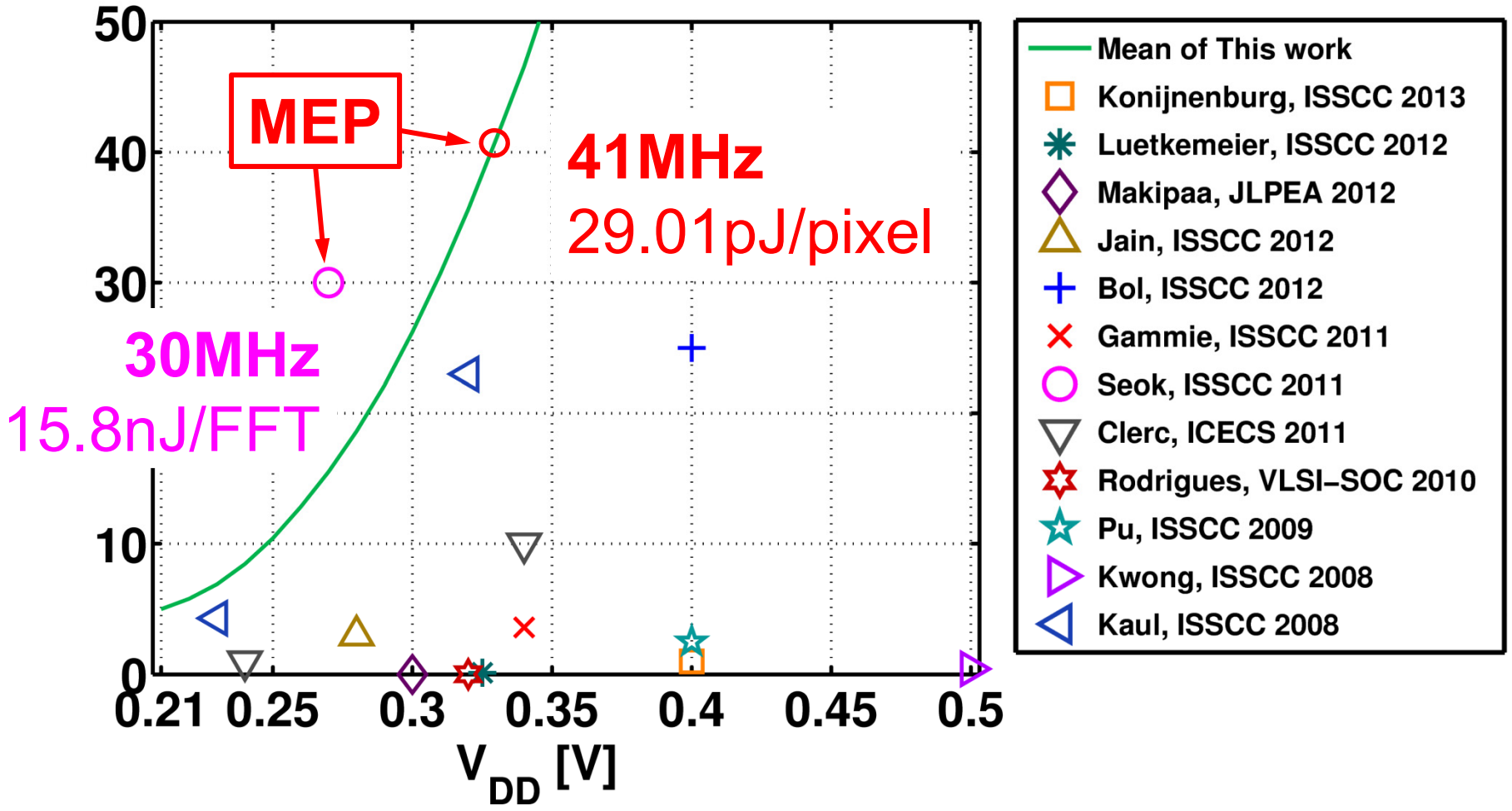
[6] Pu, ISSCC 2009: 4 x Engine (= 2D-DCT + QUANT)
1 x Huffman encoder

# Frequency comparison




**Comparison among all designs: in ≤ 65nm CMOS with $V_{DD}$ ≤ 500mV**

# Frequency comparison




**Comparison among all designs: in ≤ 65nm CMOS with $V_{DD}$ ≤ 500mV**

# Conclusion

- Near-threshold DSP
  - JPEG encoder in 40nm CMOS
- Fully functional down to 210mV
- State-of-the-art operating frequencies
- Very low energy consumption
- Variation-resilient
  - Measurement results of 26 dies
- Same design principles in other processors

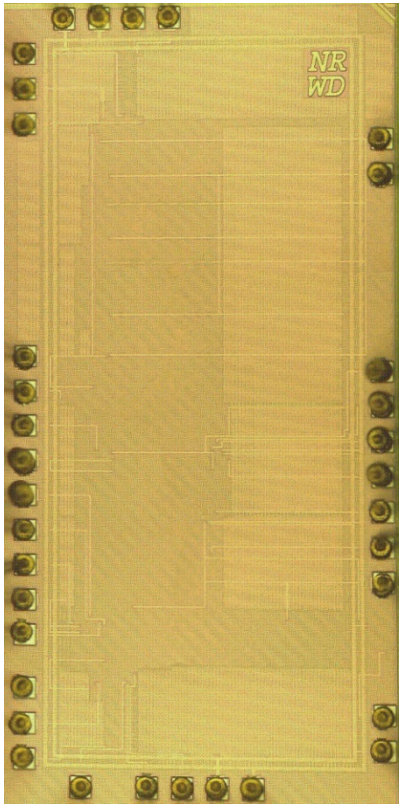
➡ Similar ultra-low-voltage characteristics

# Acknowledgement


- T. Noll, O. Weiss and M. Meixner from RWTH Aachen University for the use of Datapath Generator

- H. Reyserhove for the help with the shift registers


- **Thank you for your attention**



# A 0.75 Million-Point Fourier Transform Chip for Frequency-Sparse Signals

**Ezz El-Din Hamed**

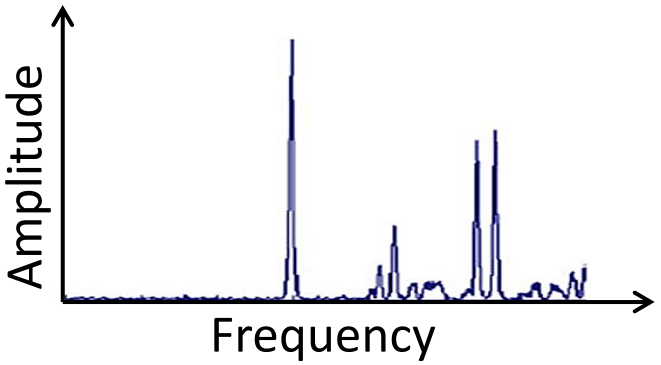
Omid Abari, Haitham Hassanieh, Abhinav Agarwal, Dina Katabi,
Anantha Chandrakasan, Vladimir Stojanović

Massachusetts Institute of Technology

wireless @mit

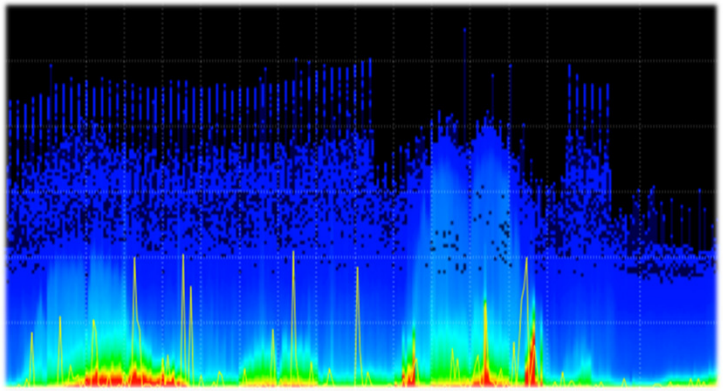
# Leverage Sparsity

- Output of FFT is Sparse:
  - Most frequencies have zero energy or noise
  - Only few frequencies are active and have energy




- Use sparse FFT algorithm
  - Find and compute the active frequencies

# Applications



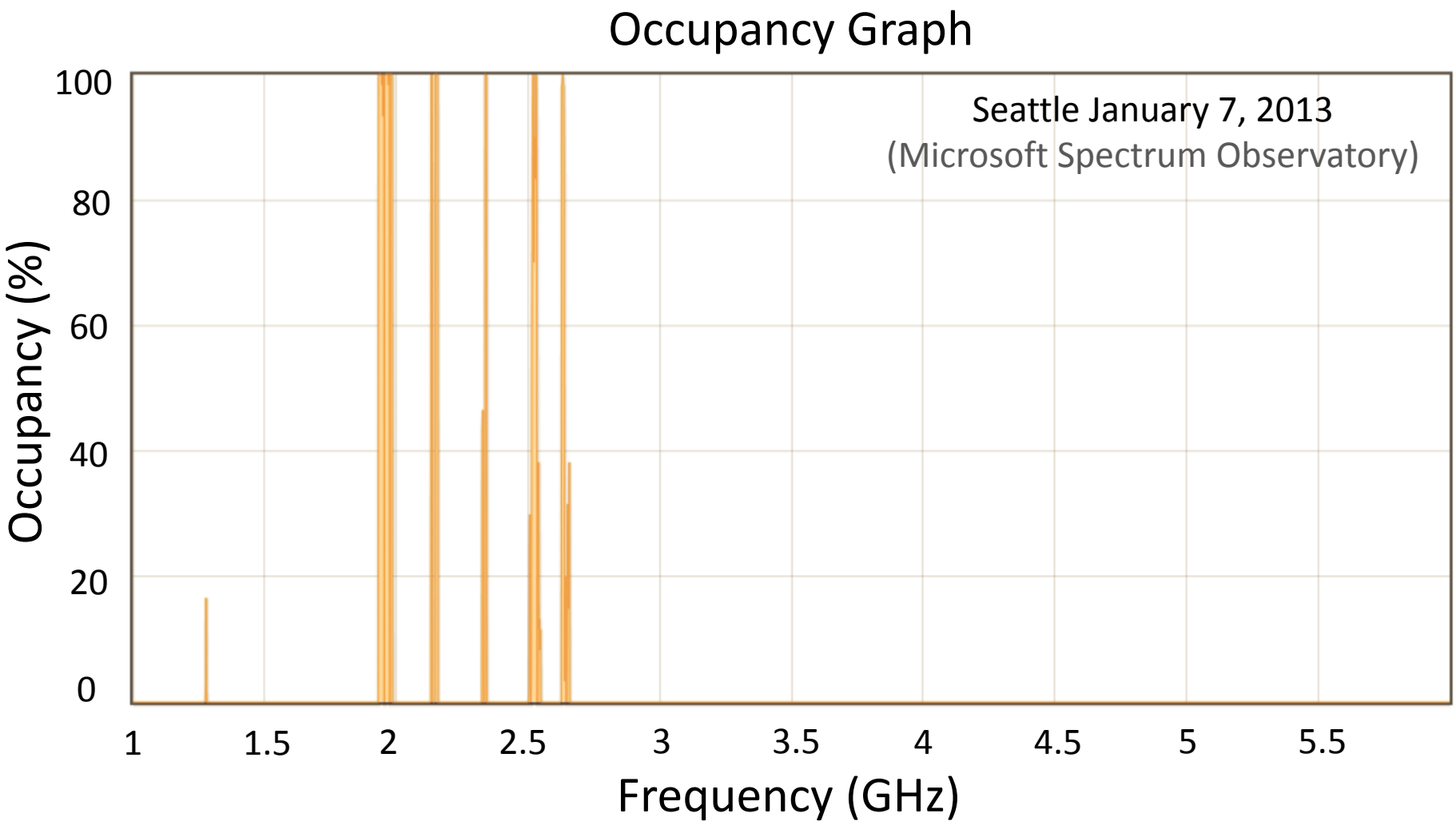
Spectrum Sensing

GPS

Radio Astronomy

- GHz spectrum sensing and acquisition
- Pattern matching by convolution with long code (e.g. GPS)
- Radio astronomy: data compression

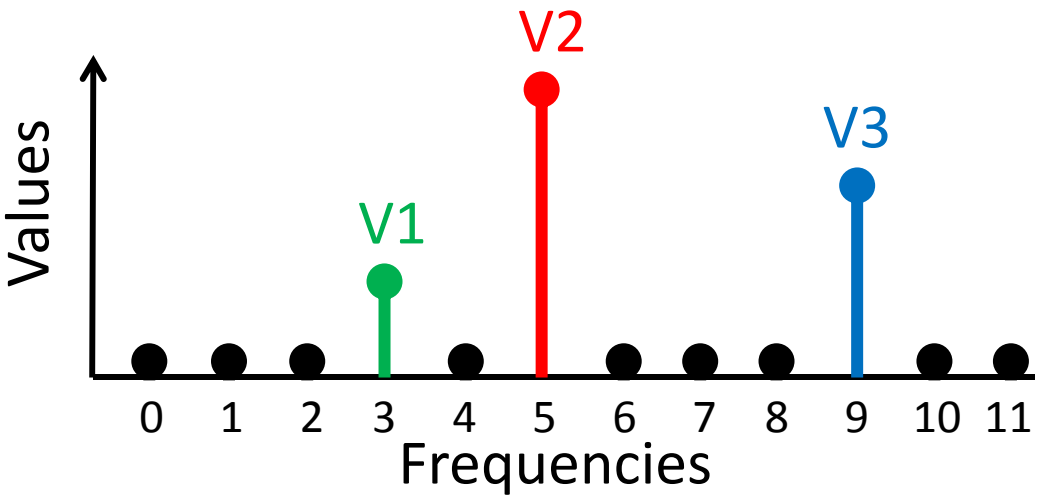
# Wideband Spectrum Sensing



# Outline

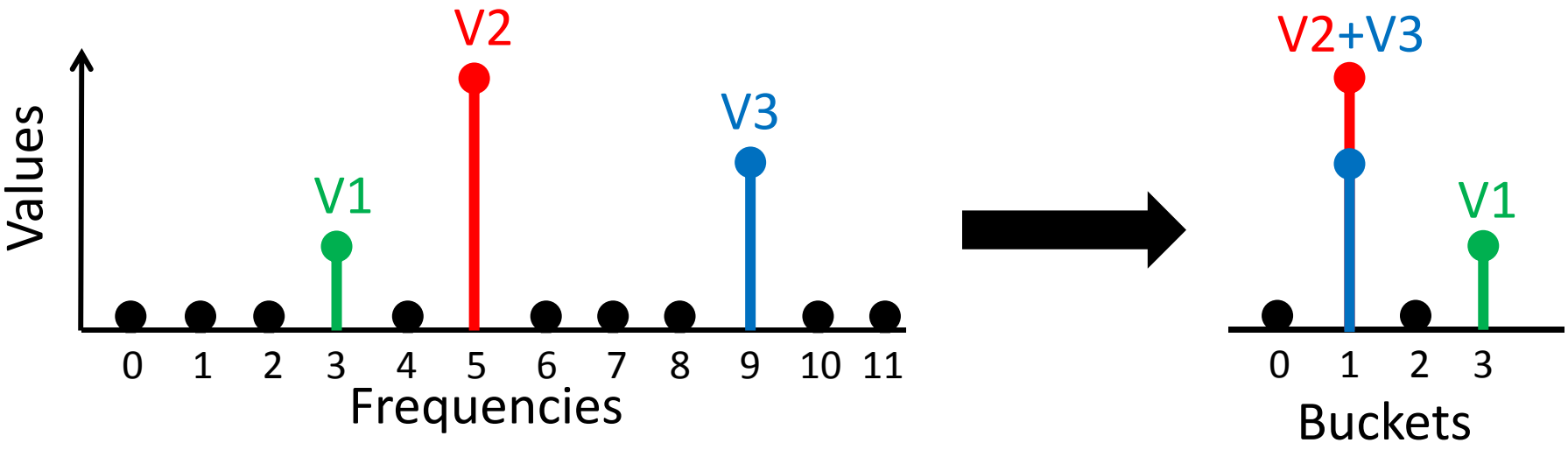
- Sparse FFT Algorithm
- Micro-architecture and Implementation
- Measurement Results
- Conclusion

# How Does Sparse FFT Work?




1. **Bucketize**
2. **Estimate**
3. **Resolve collisions**

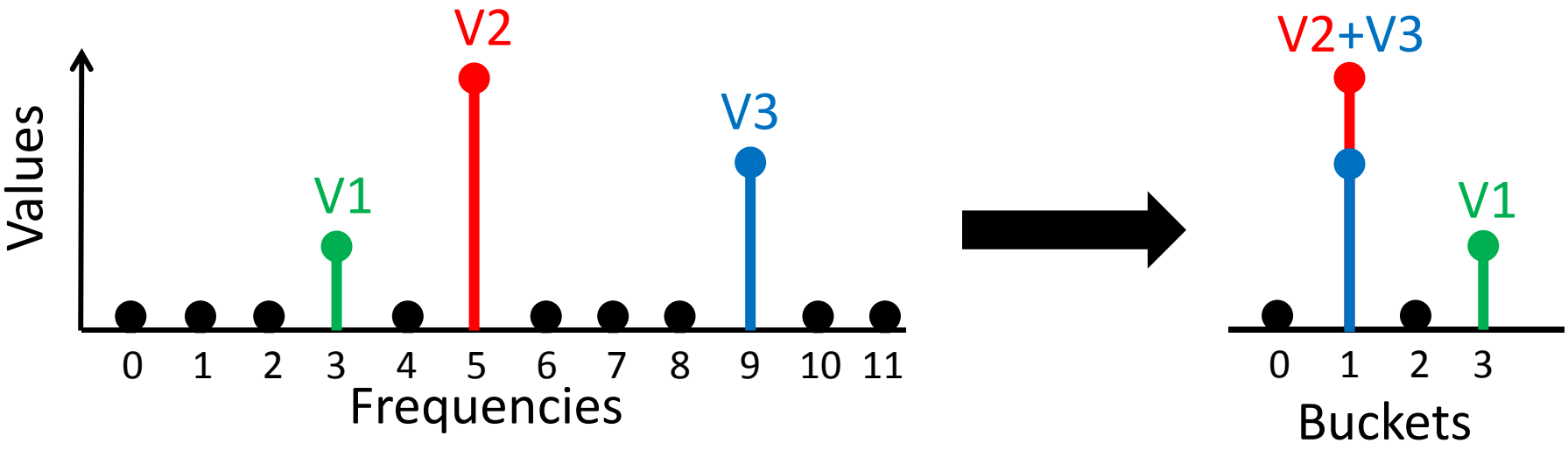
# How Does Sparse FFT Work?




## 1. Bucketize

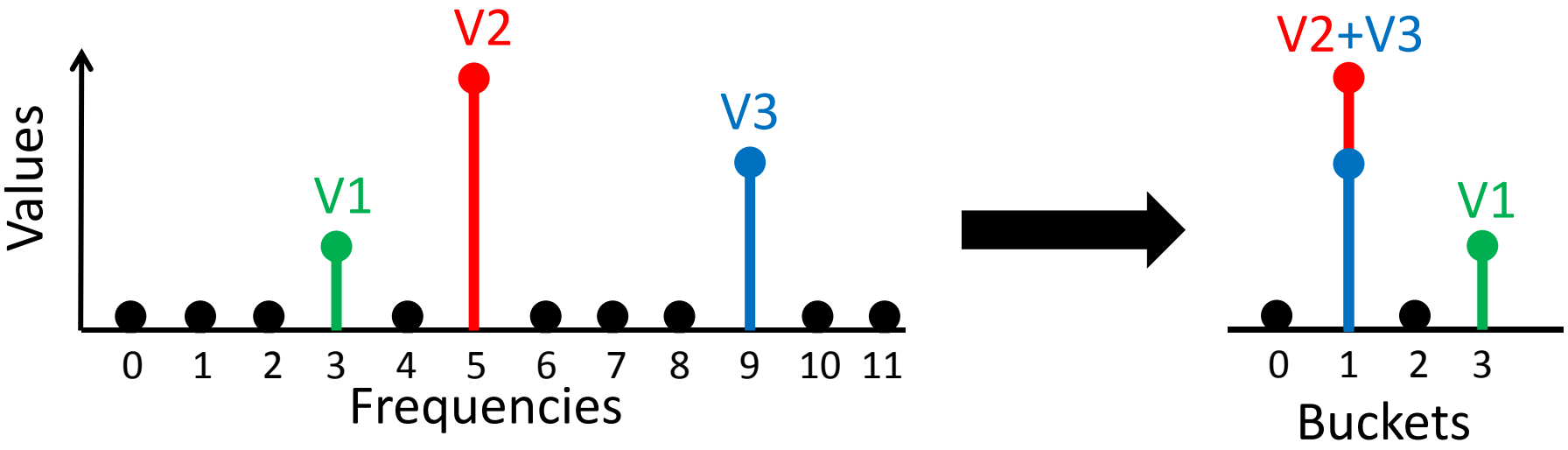
Sub-sampling time ➔ Aliasing the frequencies

# How Does Sparse FFT Work?




## 2. Estimate

# How Does Sparse FFT Work?




## 2. Estimate

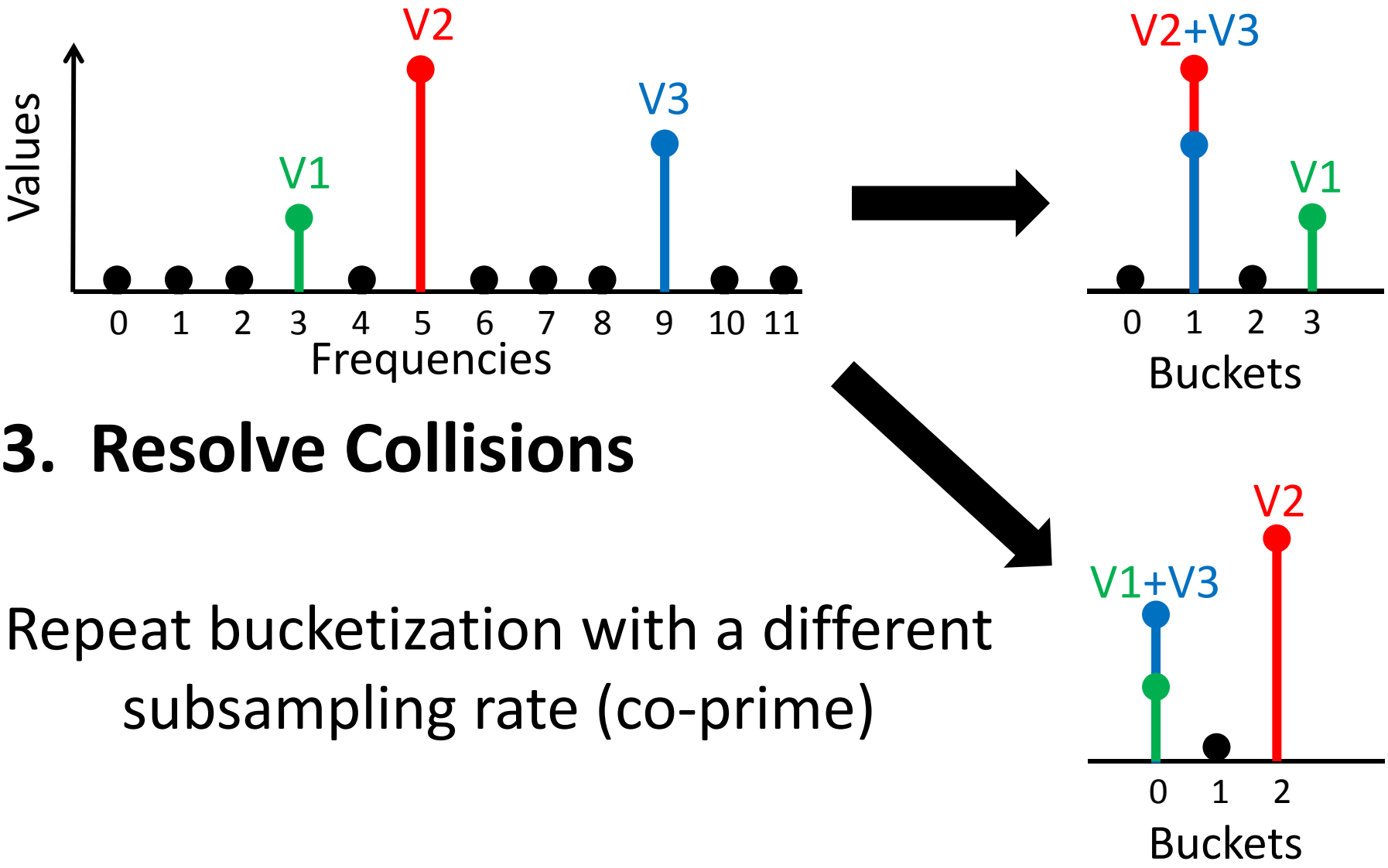
Repeat bucketization with a **time shift $\tau$**

| **Time-Domain** | **Freq-Domain** |
|---|---|
| $x(t)$ | $X(f)$ |
| $x(t-\tau)$ | $X(f)e^{-j\theta}$ |

Phase Rotation

$$\theta = \frac{2\pi f \tau}{N}$$

# How Does Sparse FFT Work?



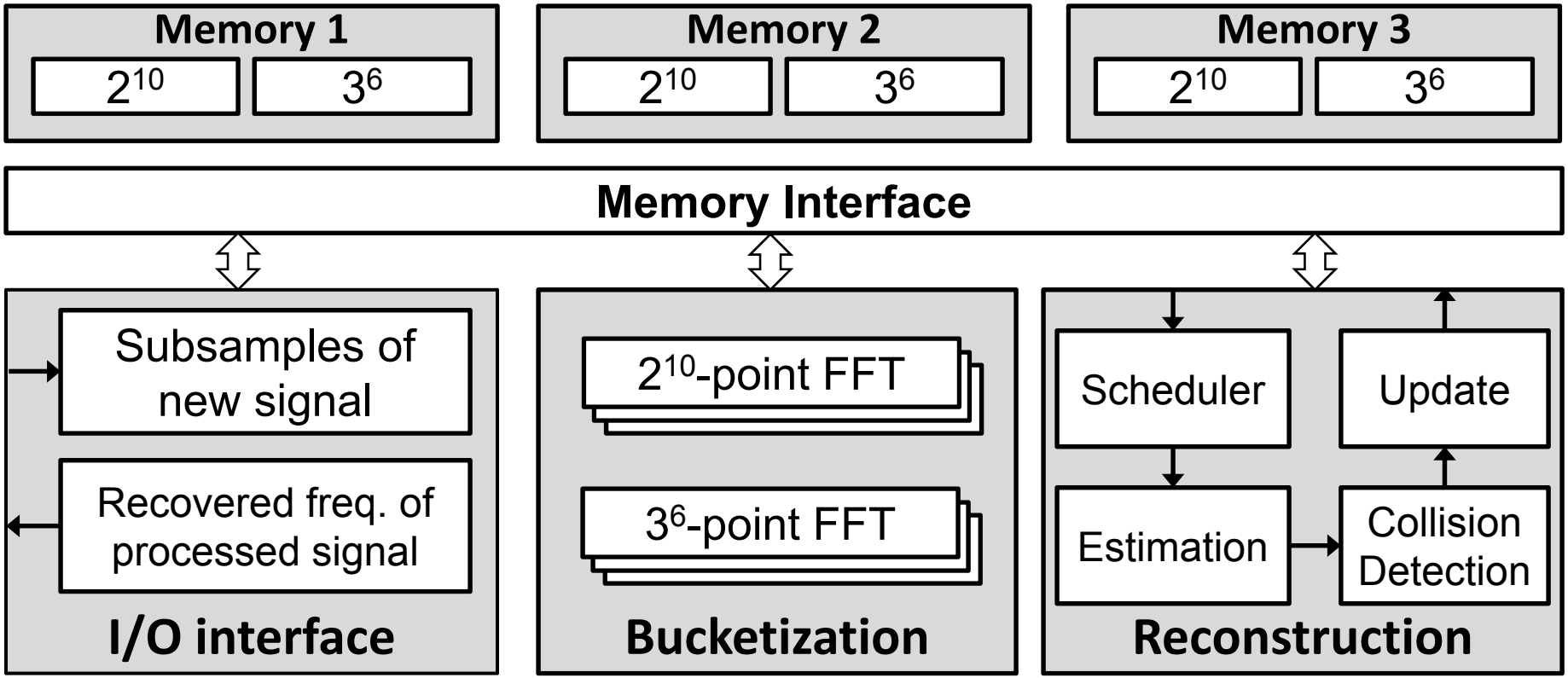

## 3. Resolve Collisions

Repeat bucketization with a different subsampling rate (co-prime)

# Implementation of Sparse FFT

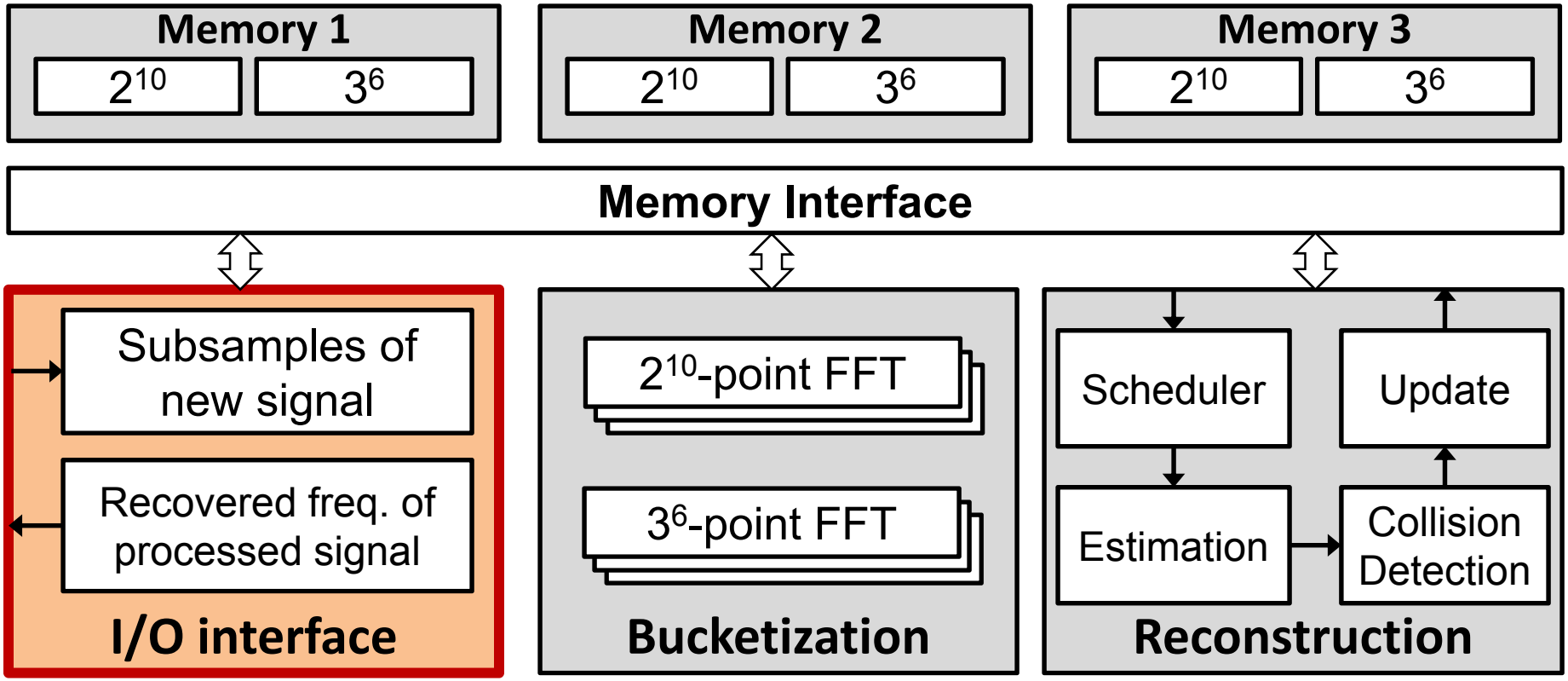
- Sparse FFT size: $N=3^6 \times 2^{10}=746496$
  - Subsample by $3^6$ in time → **FFT of size 1024 ($2^{10}$)**
  - Subsample by $2^{10}$ in time → **FFT of size 729($3^6$)**
- Sparsity: outputs up to **750** active frequencies
- Output resolution: **12 bit** real, **12 bit** imaginary

# Sparse FFT Chip Block Diagram



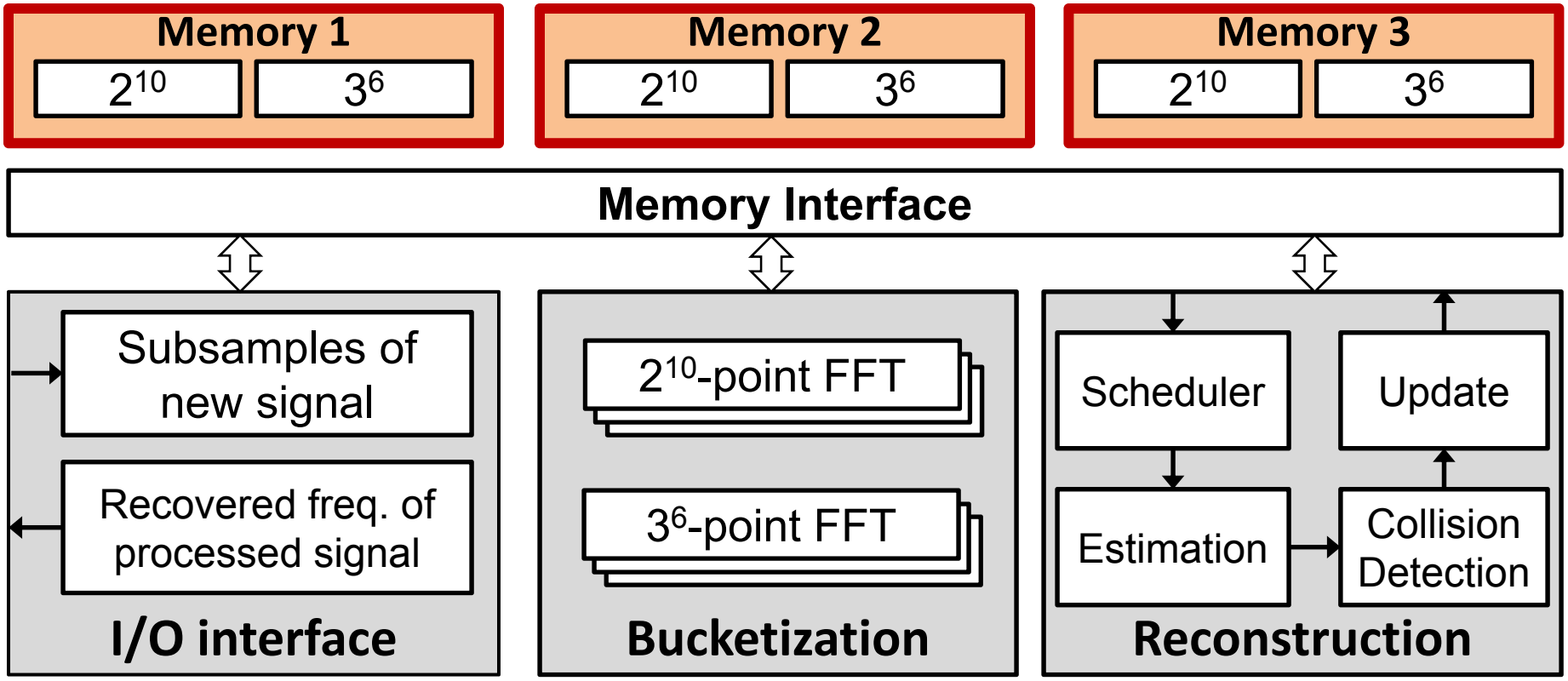

**3 Pipelined Stages**

# Sparse FFT Chip Block Diagram



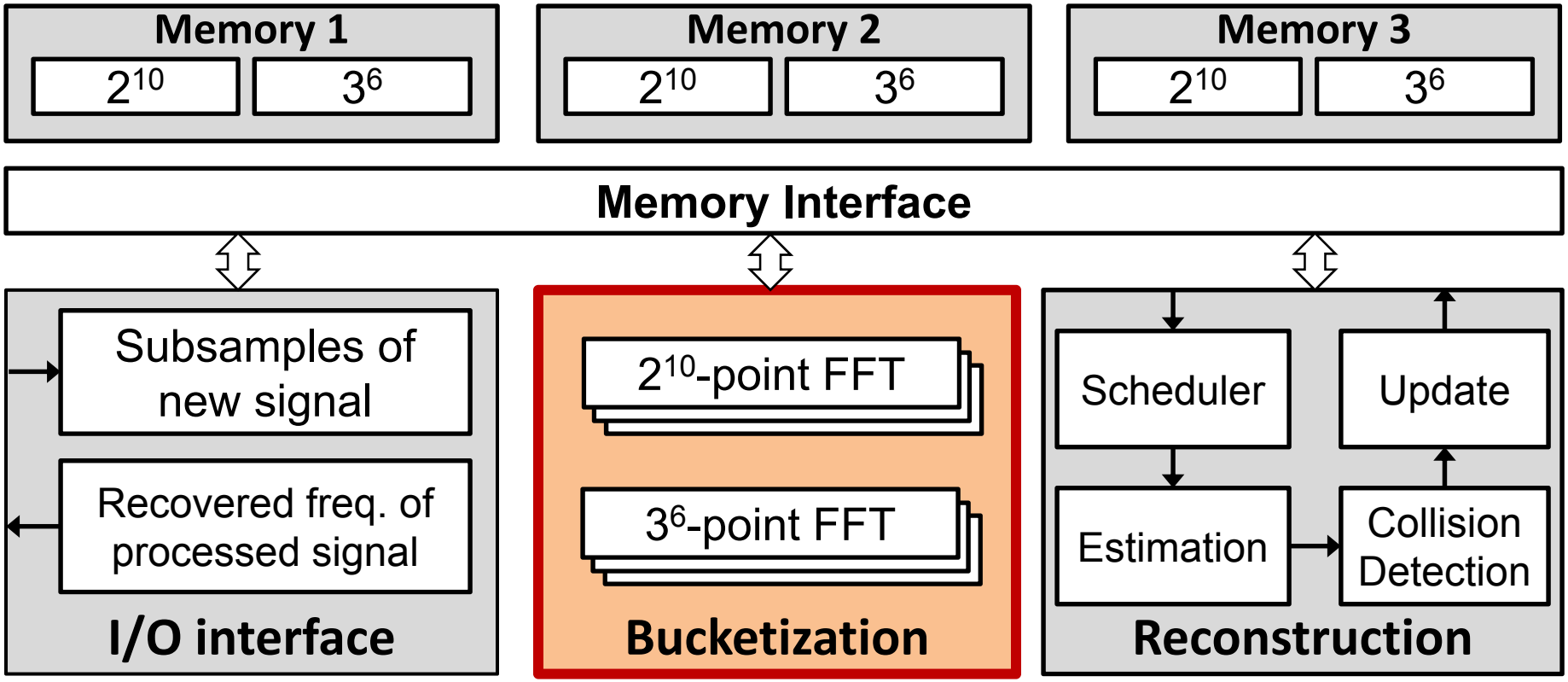

- **Input:** sub-sampled signal (by $2^{10}$ and $3^6$ )
- **Output**: positions and complex values of active frequencies

# Sparse FFT Chip Block Diagram



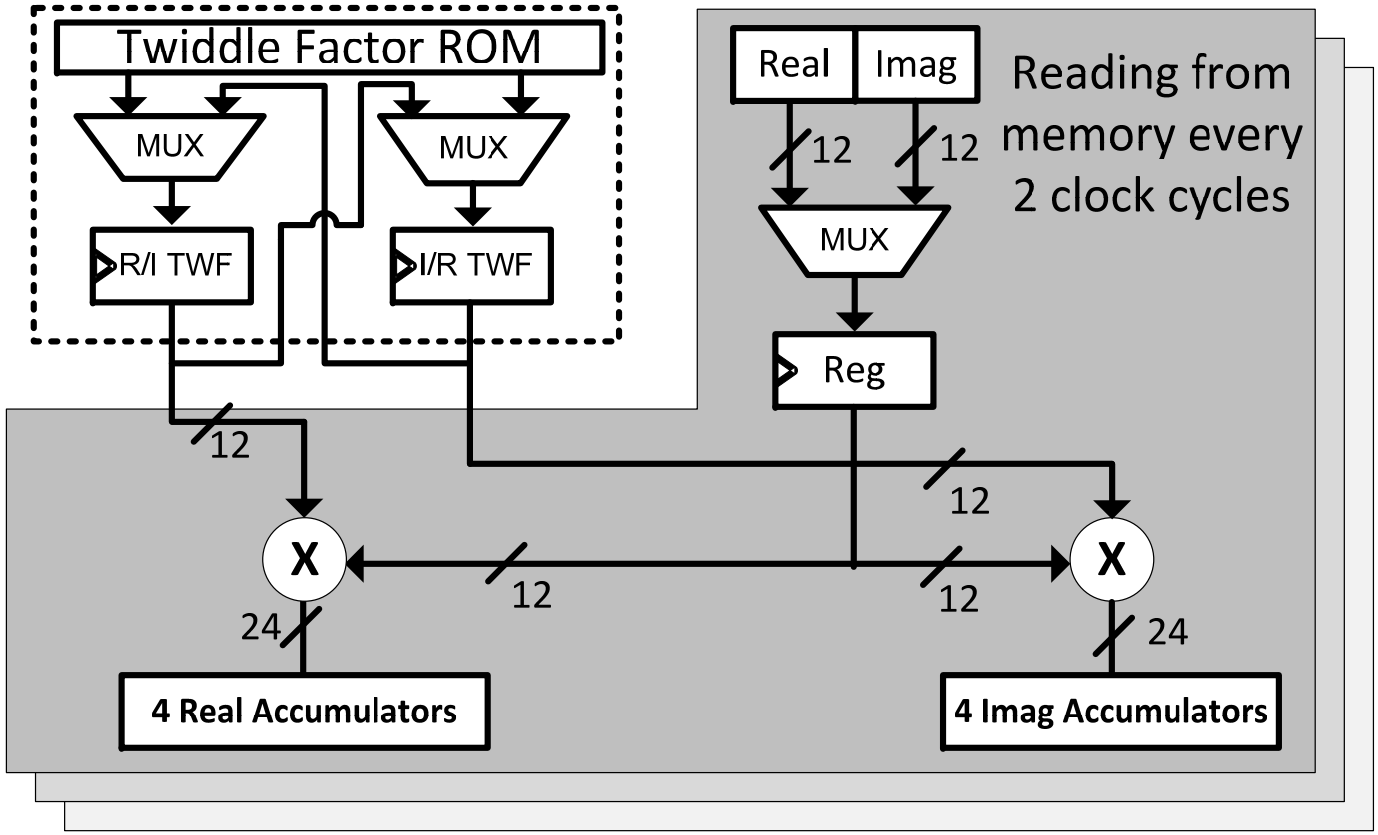

- 3 Memory sets, 2 SRAM blocks each
- Rotated across the 3 pipeline stages

# Sparse FFT Chip Block Diagram



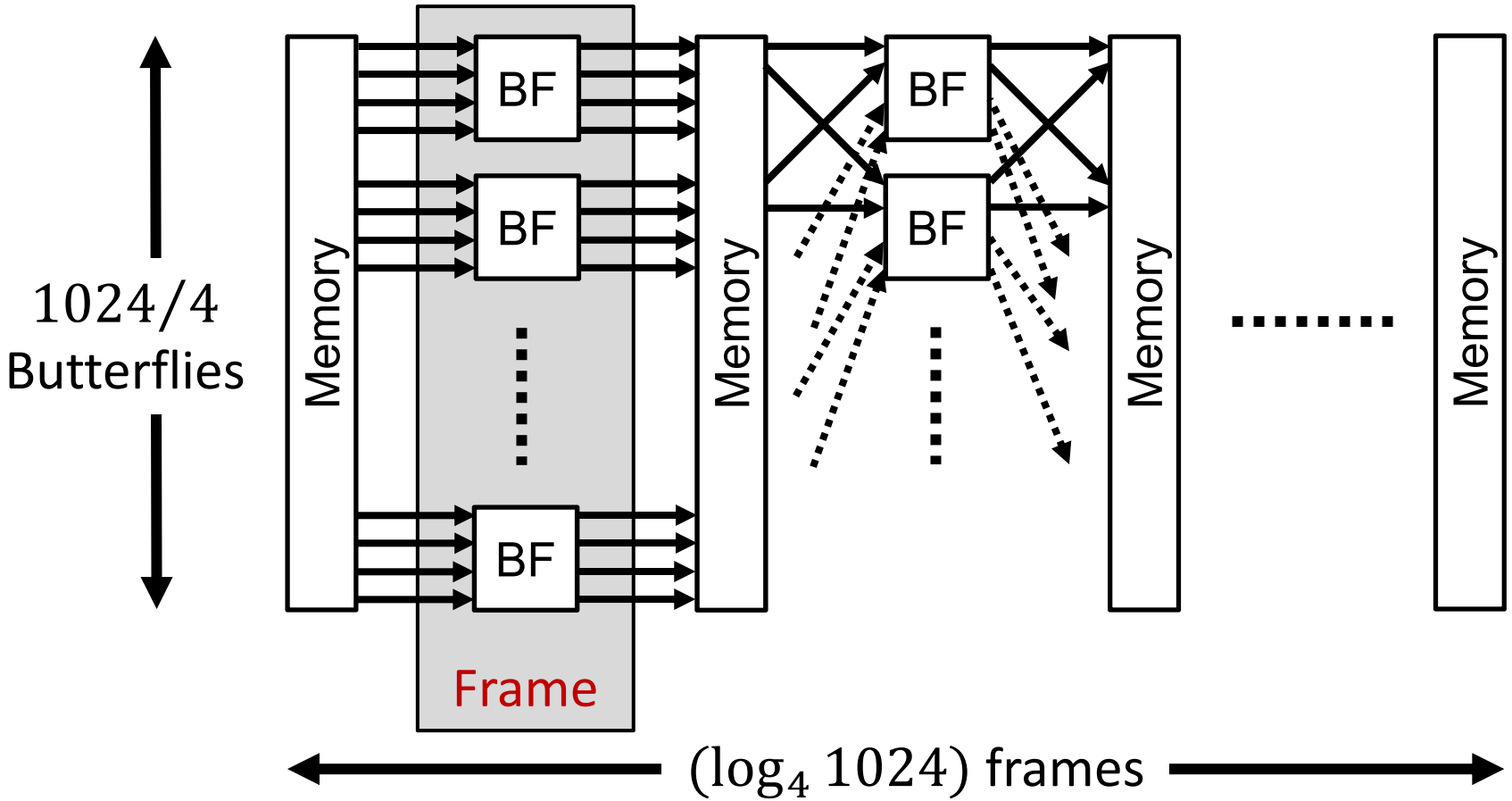

- In-place FFT
  - Radix-4 butterfly for $2^{10}$-point FFT
  - Radix-3 butterfly for $3^6$-point FFT
- Three FFTs of each type sharing the memory

# $2^{10}$ FFT Architecture



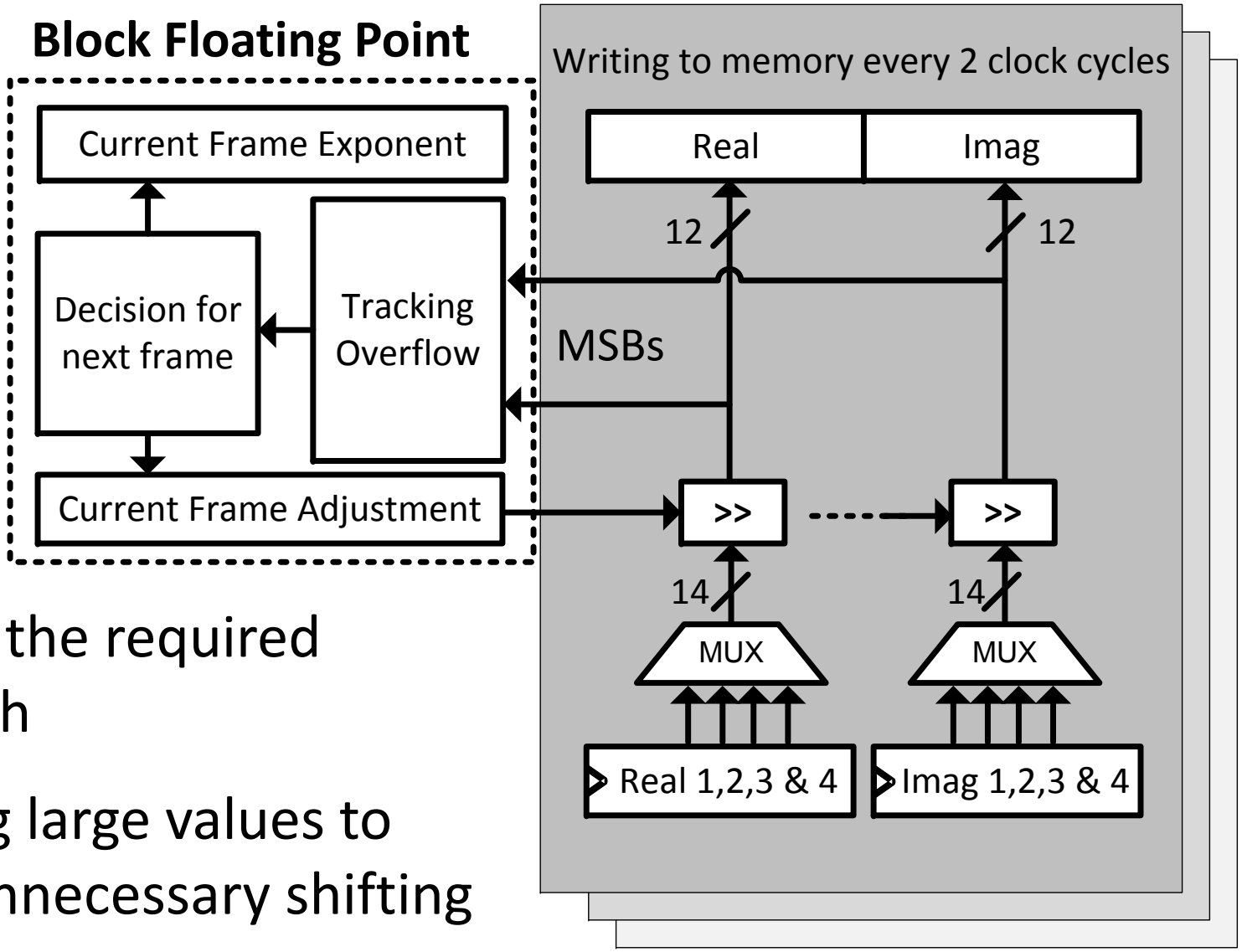

- Three radix-4 butterflies running in parallel
- Two 12b x 12b real multipliers per butterfly
- Multiplexing Real and Imaginary

# $2^{10}$ FFT Architecture



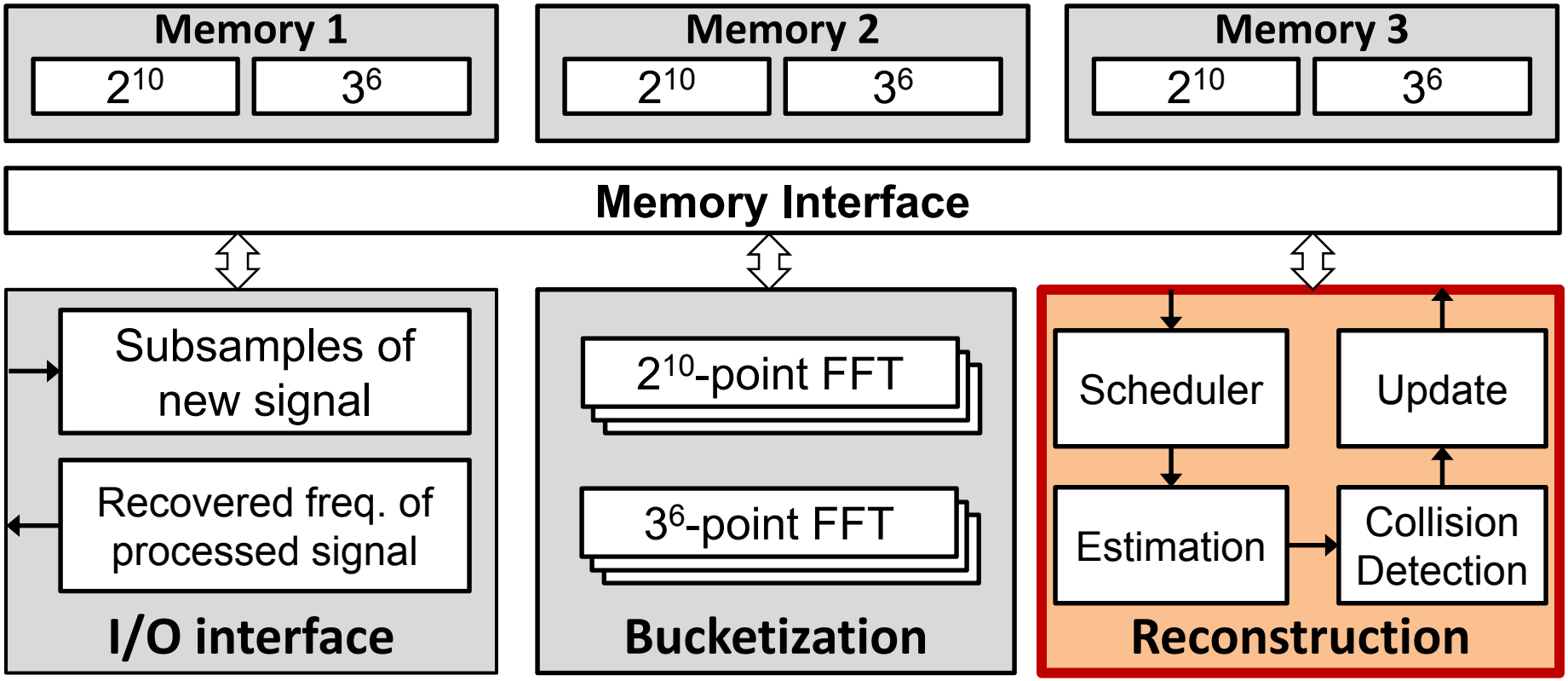

- In-place 1024-point FFT using a radix-4 butterfly
- 1024 additions (10 extra bits to account for overflow)

# Dynamic Scaling



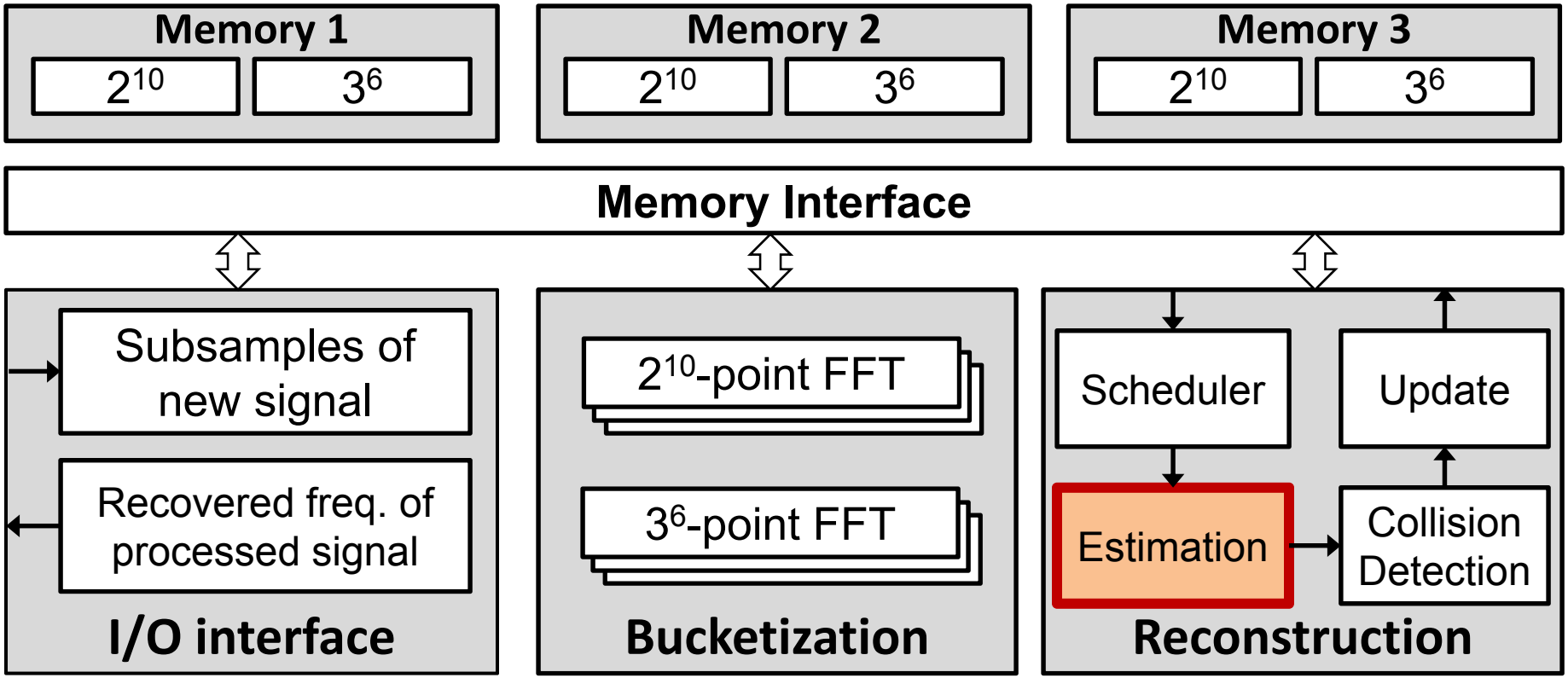

- Reduce the required bit width
- Tracking large values to avoid unnecessary shifting

# Sparse FFT Chip Block Diagram




Reconstruction stage has 4 sub-blocks

# Sparse FFT Chip Block Diagram



# Estimation Sub-block

- Estimate frequency position $f$ where $0 \leq f \leq N-1$
- For $N \approx 0.75$ million ➔ Use 20 bits to represent $f$

# Estimation Sub-block

- Estimate frequency position $f$ where $0 \leq f \leq N-1$
- For $N \approx 0.75$ million ➔ Use 20 bits to represent $f$
- For $2^{10}$ bucketization: frequency $f$ aliases to bucket number $b$:

$$b = f \bmod 2^{10}$$

10 LSBs of $f$ can be estimated from bucket number $b$

10 remaining MSBs are estimated using phase rotation

# Estimation Sub-block

Use phase rotation from a time shift $\tau$ to estimate the 10 MSBs of $f$ :

$$\theta = \frac{2\pi f \tau}{N} \quad \Longrightarrow \quad f = \frac{N\theta}{2\pi\tau}$$

# Estimation Sub-block

Use phase rotation from a time shift $\tau$ to estimate the 10 MSBs of $f$ :

$$\theta = \frac{2\pi f \tau}{N} \quad \Rightarrow \quad f = \frac{N\theta}{2\pi\tau}$$

If $\tau > 1$, phase wraps around $2\pi$

$\Downarrow$

$f_1 \neq f_2$ create same phase rotation

→loose highest MSBs

$\Downarrow$

**Use time shift $\tau = 1$ to get MSBs**

# Estimation Sub-block

Use phase rotation from a time shift $\tau$ to estimate the 10 MSBs of $f$ :

$$\theta = \frac{2\pi f \tau}{N} \quad \Rightarrow \quad f = \frac{N\theta}{2\pi\tau}$$

If $\tau > 1$, phase wraps around $2\pi$

↓

$f_1 \neq f_2$ create same phase rotation

→loose highest MSBs

↓

**Use time shift $\tau = 1$ to get MSBs**

Noise in $\theta$ creates error

↓

**Use large time shift $\tau$ to average noise in $\theta$**

# Estimation Sub-block

Use phase rotation from a time shift $\tau$ to estimate the 10 MSBs of $f$ :

$$\theta = \frac{2\pi f \tau}{N} \quad \Rightarrow \quad f = \frac{N\theta}{2\pi\tau}$$

If $\tau > 1$, phase wraps around $2\pi$

↓

$f_1 \neq f_2$ create same phase rotation

→loose highest MSBs

↓

**Use time shift $\tau = 1$ to get MSBs**

↓

Noise in $\theta$ creates error

↓

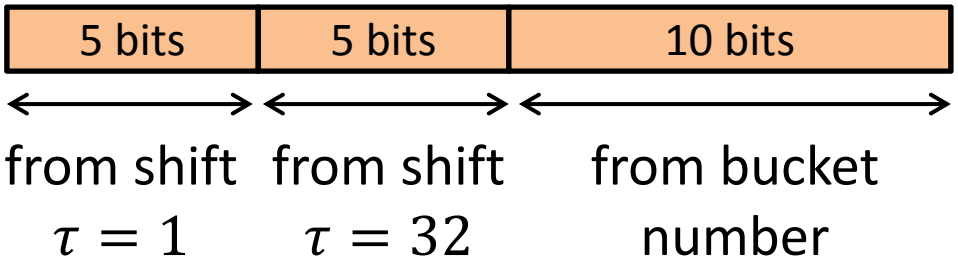
**Use large time shift $\tau$ to average noise in $\theta$**

**Use two different time shifts**
**$\tau = 1$ and $\tau = 32$ to estimate the 10 MSBs of $f$**

# Estimation Sub-block

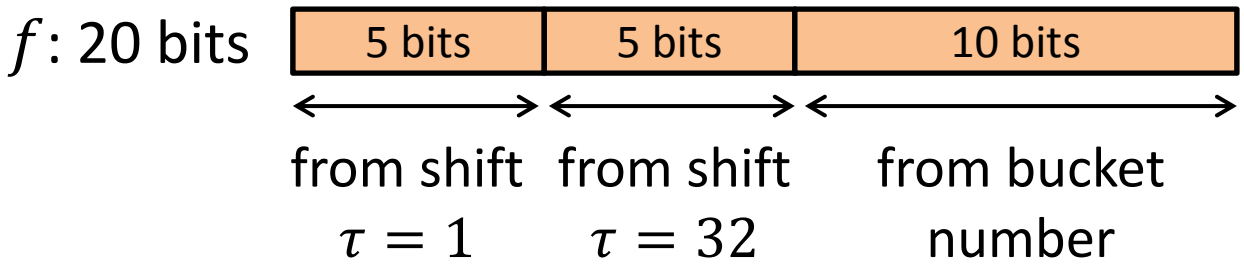
Frequency Recovery from $2^{10}$ bucketization

$f$: 20 bits



# Estimation Sub-block

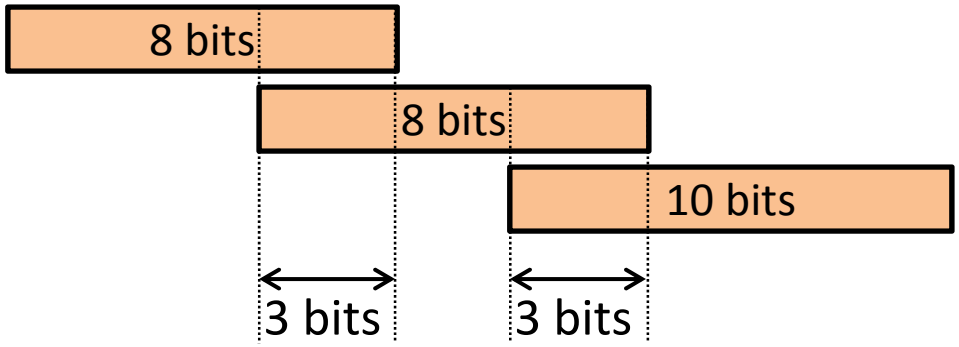
Frequency Recovery from $2^{10}$ bucketization



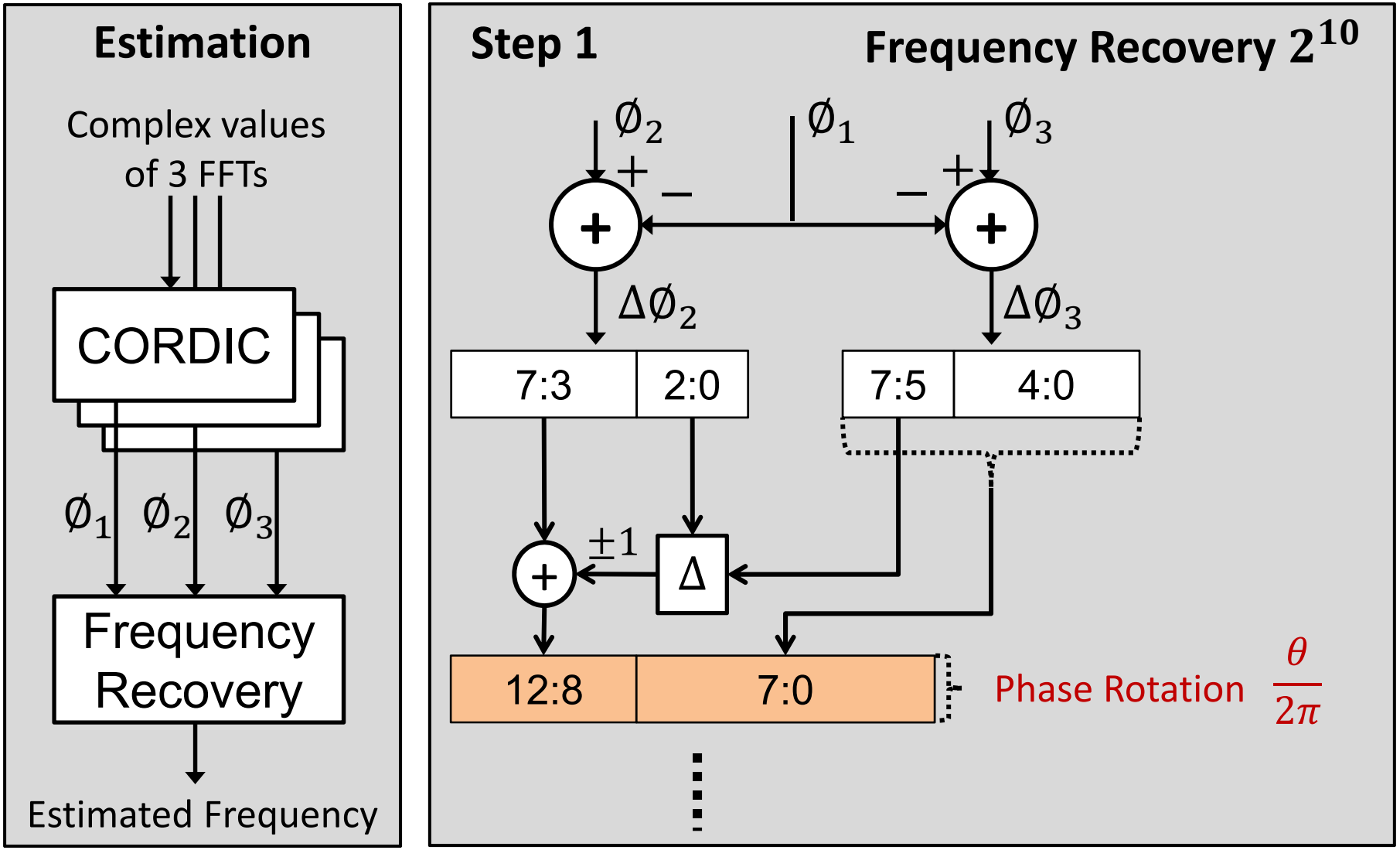

**Problem:** Small error propagates

→ Affect MSBs

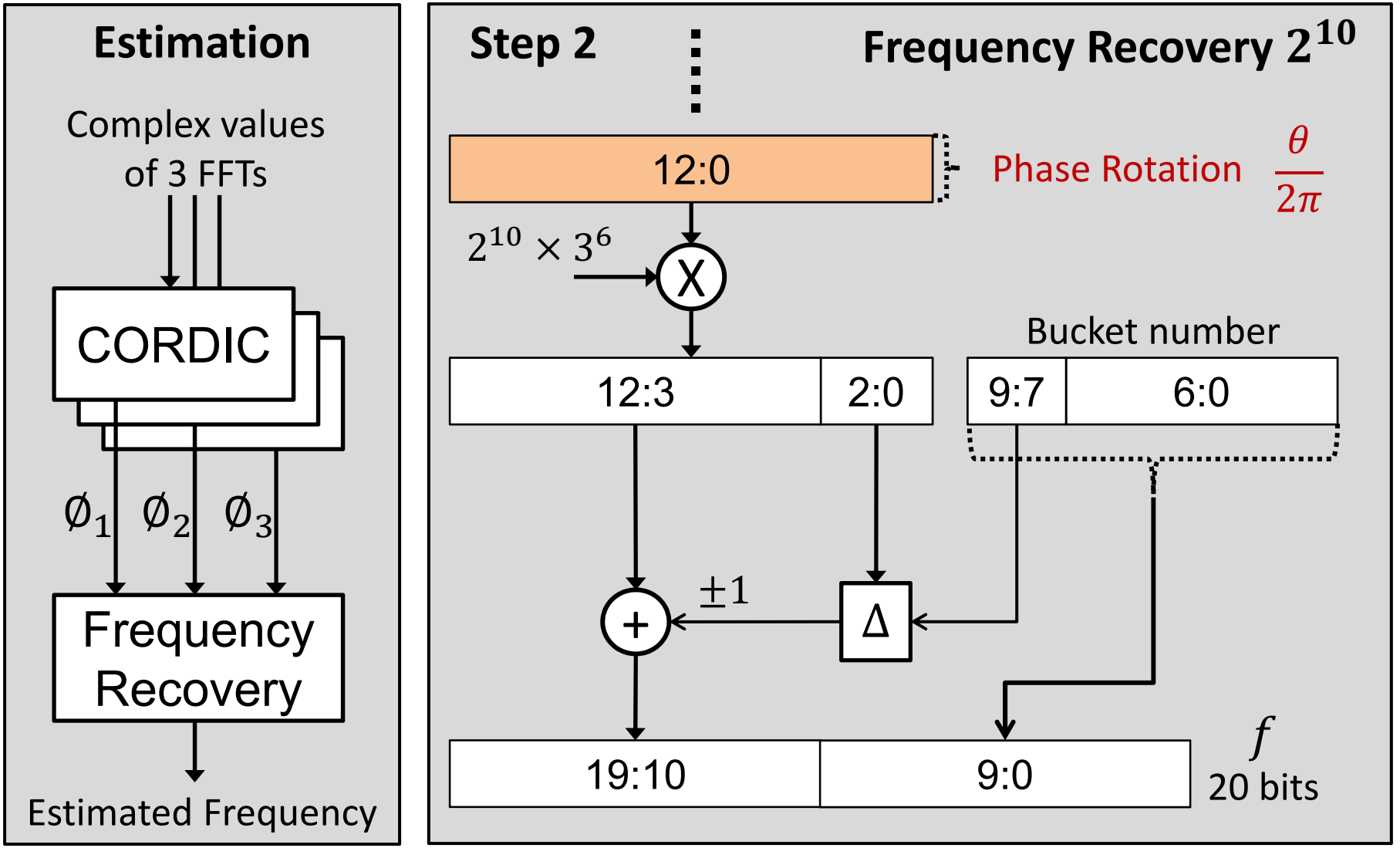
**Solution:** Use 3 bit overlap to detect errors



# Estimation Sub-block



# Estimation Sub-block



# Estimation Sub-block

- For $3^6$ bucketization: frequency $f$ aliases to bucket number $b$:

$$b = f \bmod 3^6$$

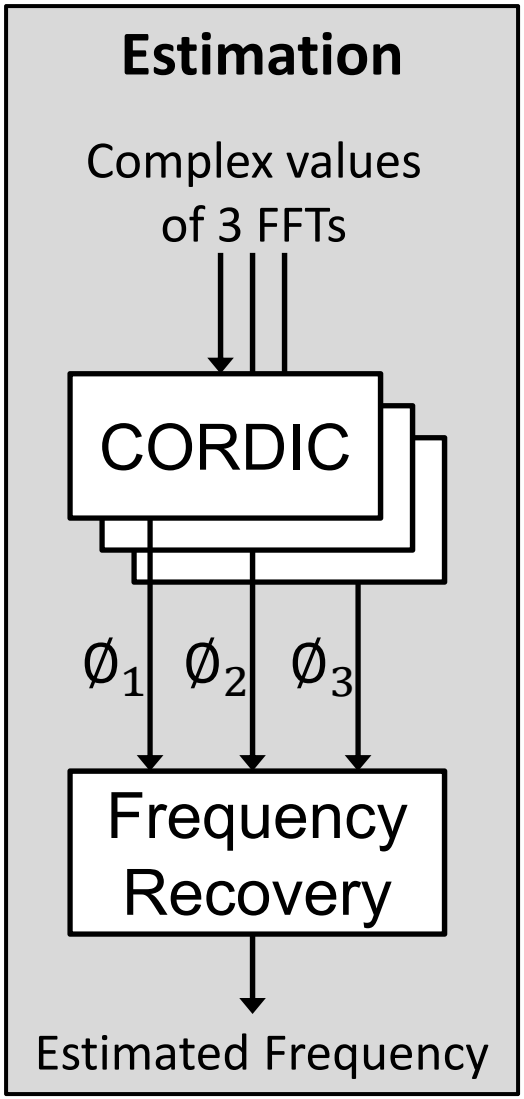
- LSBs cannot be replaced directly by the bucket number

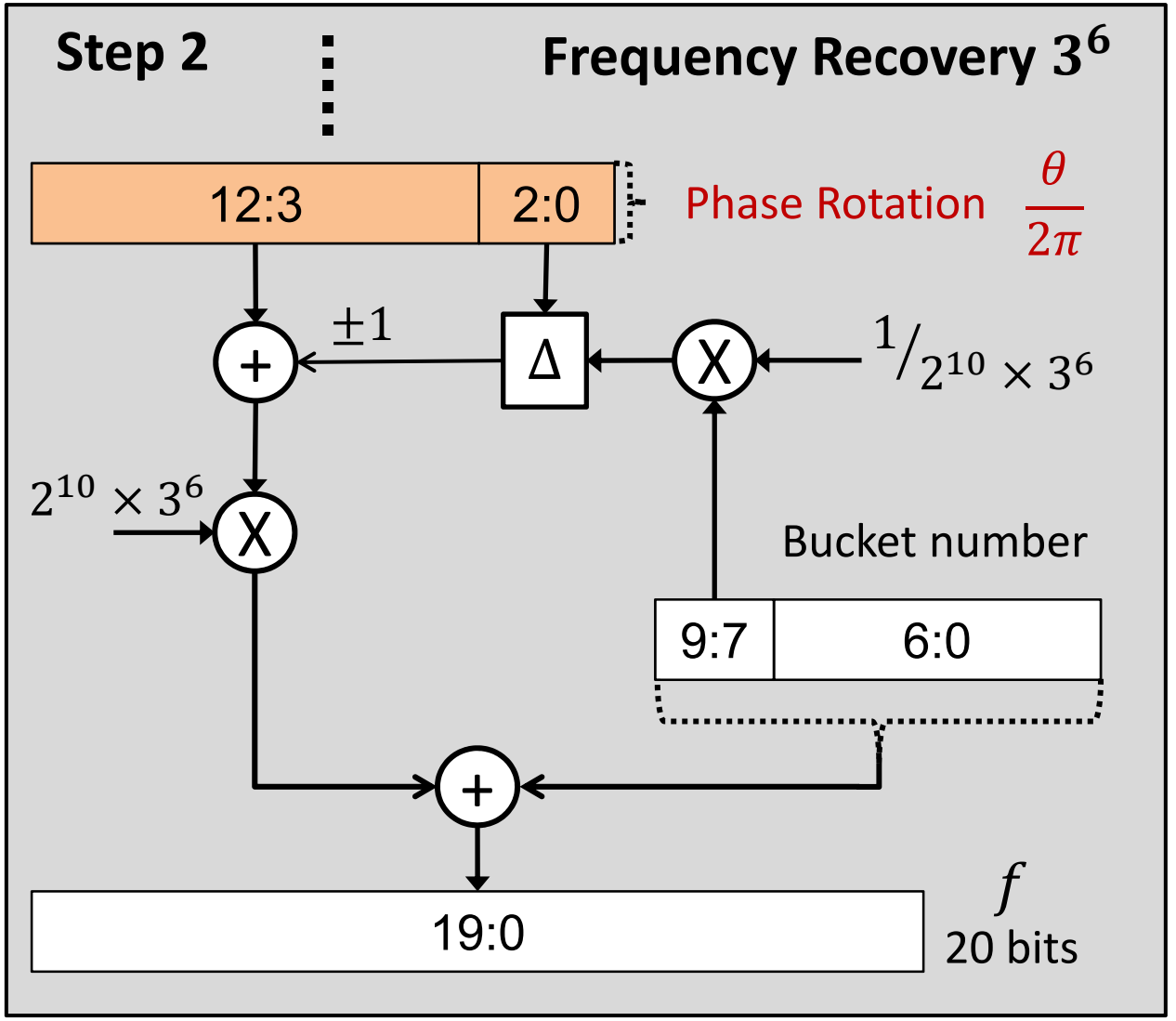
- Naïve implementation:
  1. Estimate $f$ from the phase rotation
  2. Calculate the remainder $(f \bmod 3^6)$
  3. Subtract this remainder from $f$
  4. Then add the bucket number $b$

$$f = \frac{N\theta}{2\pi\tau}$$

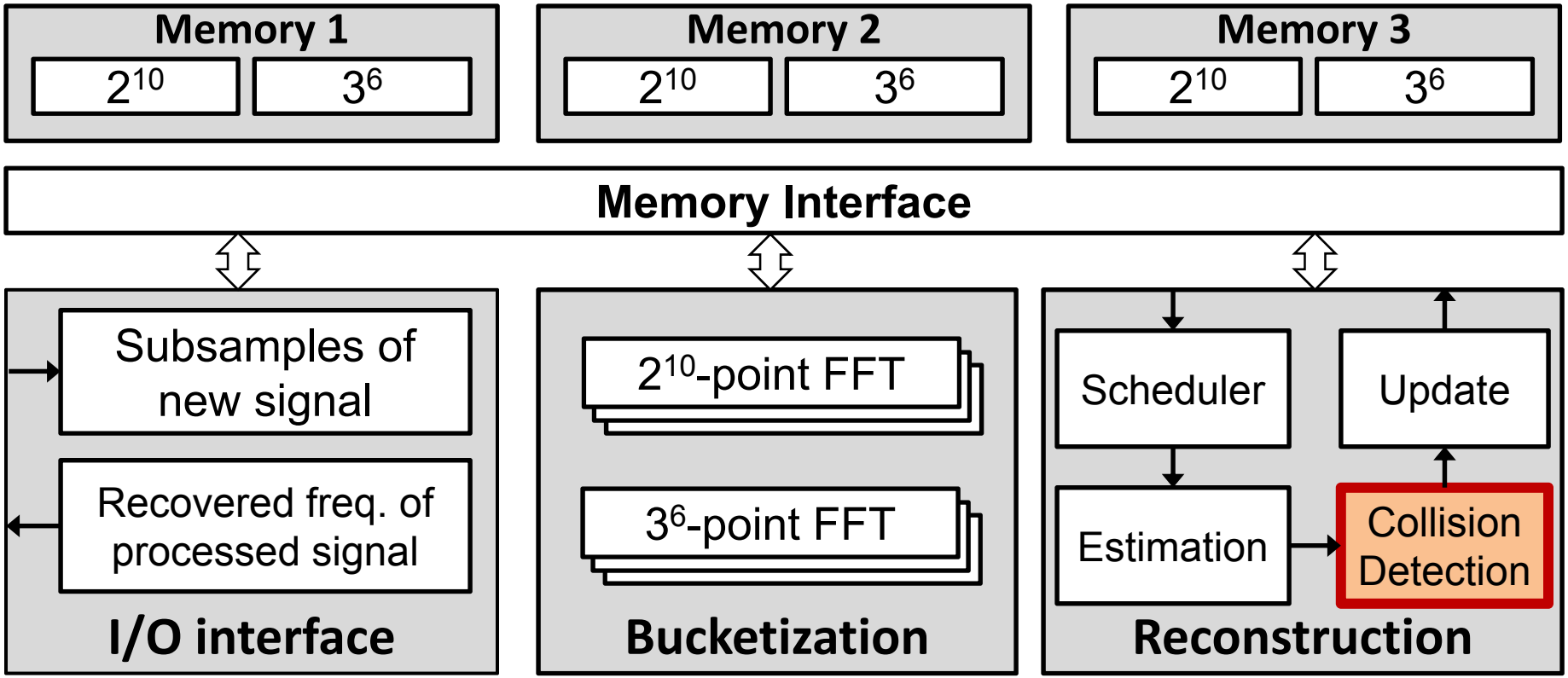
- Steps 2 and 3 are done indirectly by truncating the LSBs of $\theta$

# Estimation Sub-block



# Sparse FFT Chip Block Diagram



# Collision Detection

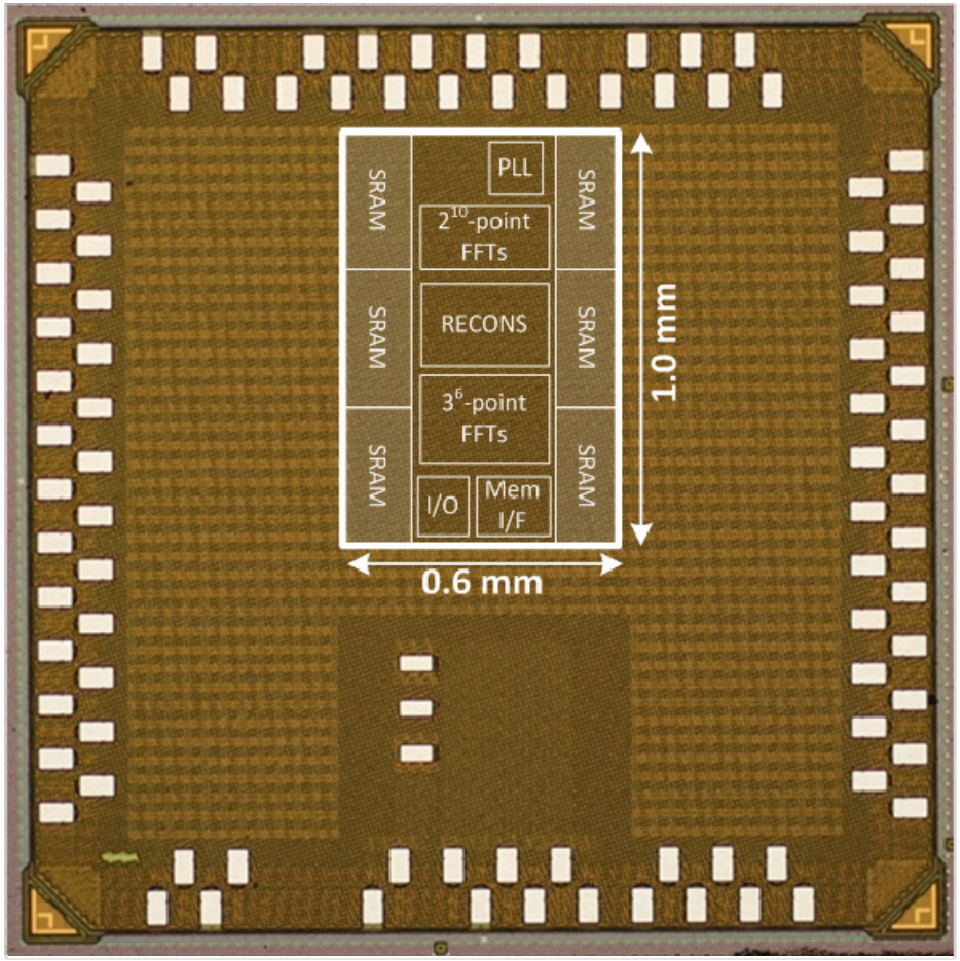
- No Collision:
  - Single active frequency
  - Time shift causes only change in phase and not in magnitude of the bucket

# Collision Detection

- No Collision:
  - Single active frequency
  - Time shift causes only change in phase and not in magnitude of the bucket

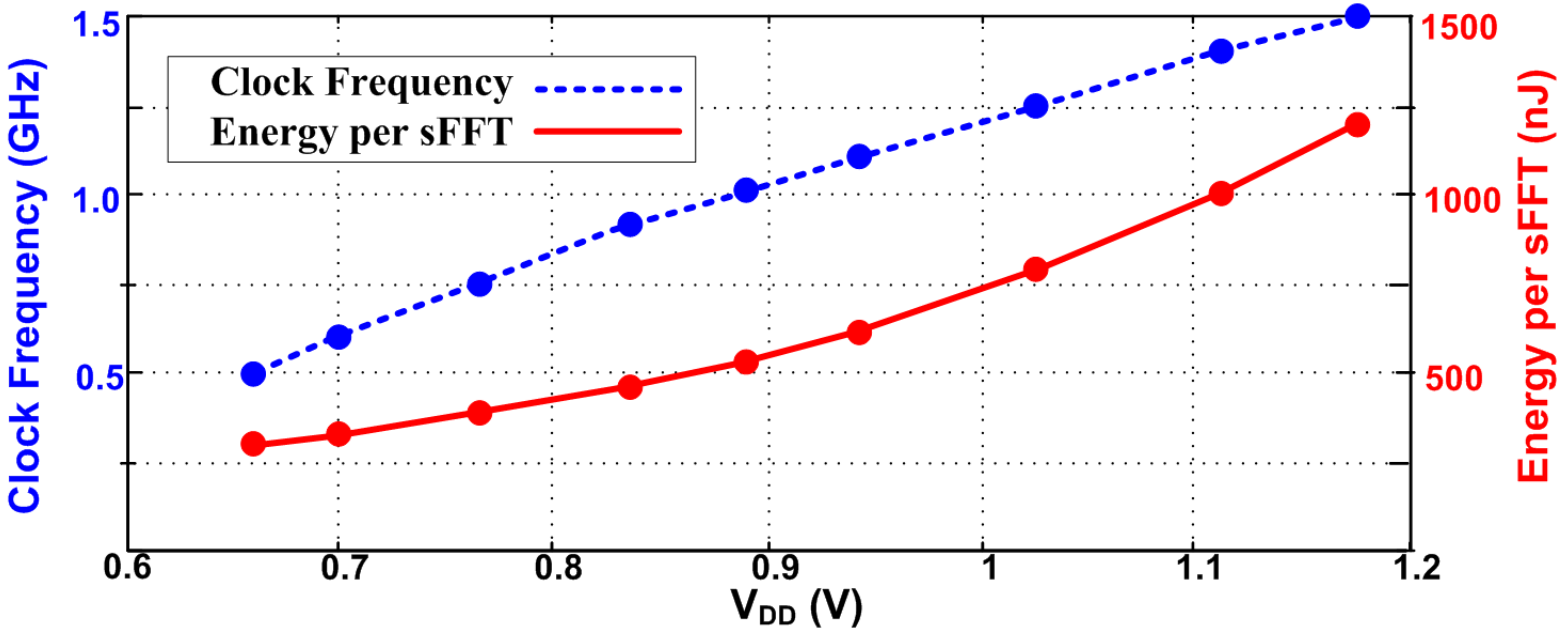
- Collision:
  - Multiple active frequencies sum up with different phase rotations
  - Time shift causes the magnitude of the bucket to change

# Die photo

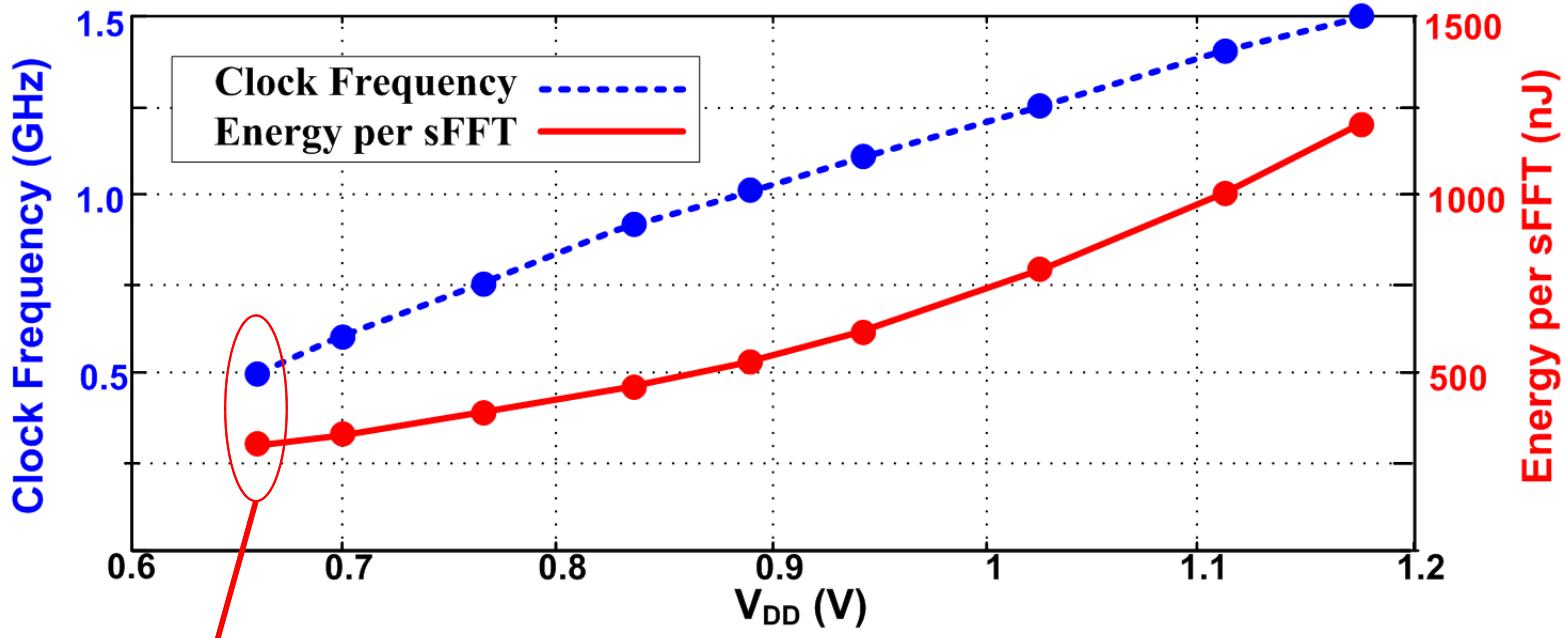



| Technology | 45nm SOI CMOS |
|---|---|
| Core Area | 0.6mm x 1.0mm |
| Core Supply Voltage | 0.66-1.18 V |
| Clock Frequency | 0.5-1.5 GHz |
| Core Power | 14.6 - 174.8 mW |
| Latency | 63 - 21 µs |

# Measurement Result

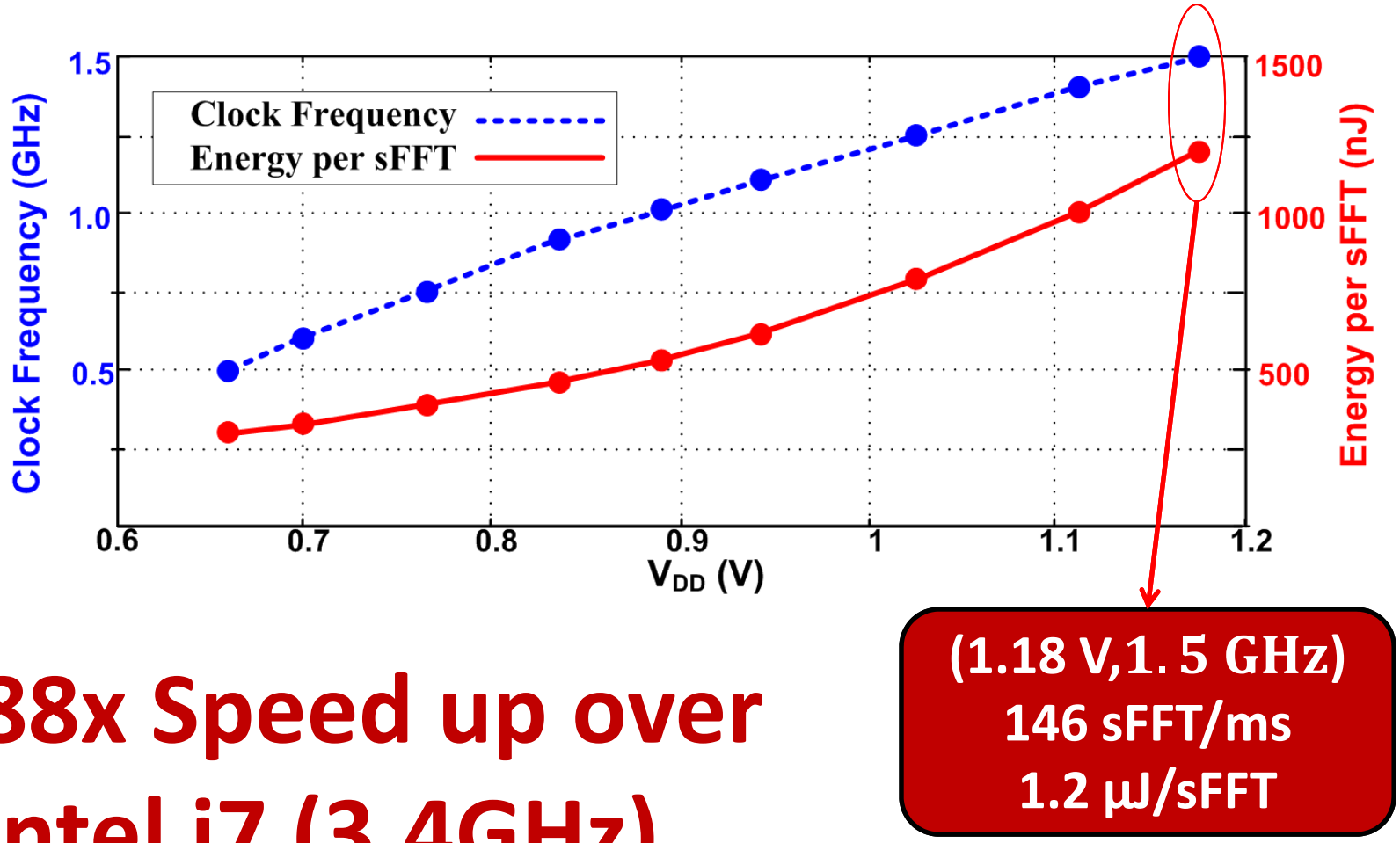


# Measurement Result



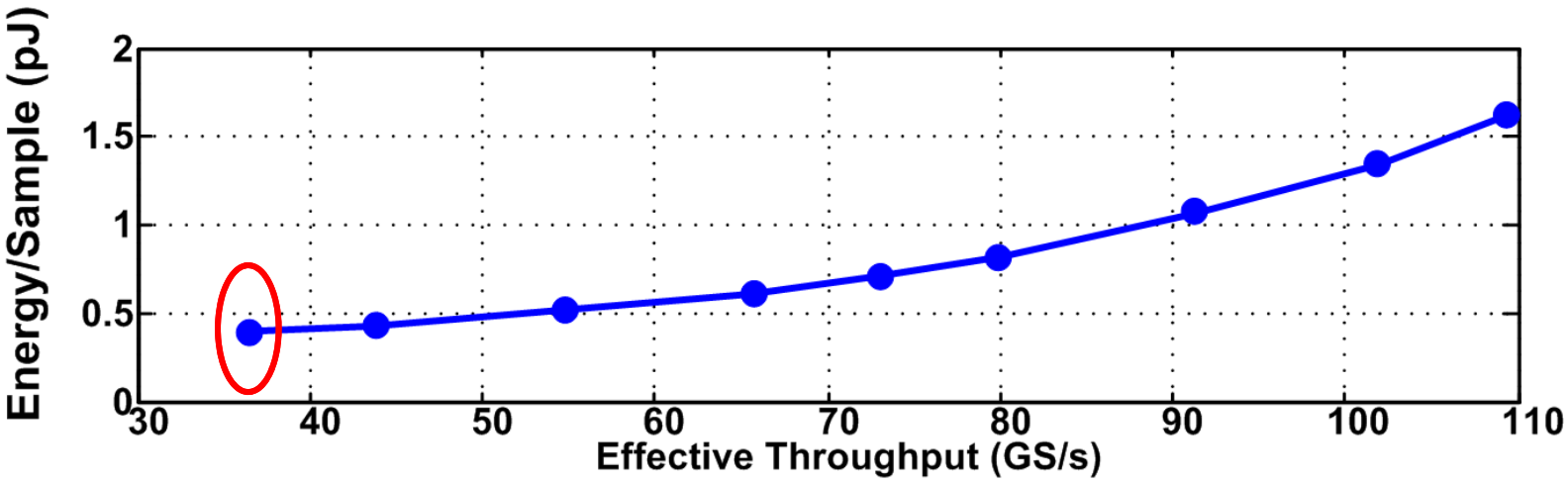

**(0.66 V, 500 MHz)**
**48 sFFT/ms**
**298 nJ/sFFT**

# Measurement Result




**88x Speed up over intel i7 (3.4GHz)**

# Measurement Result




At 0.66V:

- Effective throughput 36GS/s
- Consuming 0.4pJ/sample

# Comparison

| | ISSCC'11 | JSSC'08 | JSSC'12 | **This work** |
|---|---|---|---|---|
| Technology | 65 nm | 90 nm | 65nm | 45 nm |
| Signal Type | Any Signal | Any Signal | Any Signal | Freq.-Sparse signal |
| Size | $2^{10}$ | $2^{8}$ | $2^{7}$ to $2^{11}$ | $3^{6}x2^{10}$ |
| Word width | 16 bits | 10 bits | 12 bits | 12 bits |
| Area | $8.29mm^2$ | $5.1mm^2$ | $1.37mm^2$ | $0.6mm^2$ |
| Effective Throughput | 240MS/s | 2.4GS/s | 1.25-20MS/s | 36-109GS/s |
| Energy/Sample | 17.2pJ | 50pJ | 19.5-50.6pJ | 0.4-1.6pJ |

# Comparison

| | ISSCC'11 | JSSC'08 | JSSC'12 | **This work** |
|---|---|---|---|---|
| Technology | 65 nm | 90 nm | 65nm | 45 nm |
| Signal Type | Any Signal | Any Signal | Any Signal | Freq.-Sparse signal |
| Size | $2^{10}$ | $2^{8}$ | $2^{7}$ to $2^{11}$ | $3^{6}x2^{10}$ |
| Word width | 16 bits | 10 bits | 12 bits | 12 bits |
| Area | 8.29mm$^2$ | 5.1mm$^2$ | 1.37mm$^2$ | 0.6mm$^2$ |
| Effective Throughput | 240MS/s | 2.4GS/s | 1.25-20MS/s | 36-109GS/s |
| Energy/Sample | 17.2pJ | 50pJ | 19.5-50.6pJ | 0.4-1.6pJ |

**Sparse FFT works only for frequency-sparse signals**

# Comparison

| | ISSCC'11 | JSSC'08 | JSSC'12 | **This work** |
|---|---|---|---|---|
| Technology | 65 nm | 90 nm | 65nm | 45 nm |
| Signal Type | Any Signal | Any Signal | Any Signal | Freq.-Sparse signal |
| Size | $2^{10}$ | $2^{8}$ | $2^{7}$ to $2^{11}$ | $3^{6}x2^{10}$ |
| Word width | 16 bits | 10 bits | 12 bits | 12 bits |
| Area | 8.29mm$^2$ | 5.1mm$^2$ | 1.37mm$^2$ | 0.6mm$^2$ |
| Effective Throughput | 240MS/s | 2.4GS/s | 1.25-20MS/s | 36-109GS/s |
| Energy/Sample | 17.2pJ | 50pJ | 19.5-50.6pJ | 0.4-1.6pJ |

# Comparison

| | ISSCC'11 | JSSC'08 | JSSC'12 | **This work** |
|---|---|---|---|---|
| Technology | 65 nm | 90 nm | 65nm | 45 nm |
| Signal Type | Any Signal | Any Signal | Any Signal | Freq.-Sparse signal |
| Size | $2^{10}$ | $2^{8}$ | $2^{7}$ to $2^{11}$ | $3^{6}x2^{10}$ |
| Word width | 16 bits | 10 bits | 12 bits | 12 bits |
| Area | 8.29mm$^2$ | 5.1mm$^2$ | 1.37mm$^2$ | 0.6mm$^2$ |
| Effective Throughput | 240MS/s | 2.4GS/s | 1.25-20MS/s | 36-109GS/s |
| Energy/Sample | 17.2pJ | 50pJ | 19.5-50.6pJ | 0.4-1.6pJ |

# Comparison

| | ISSCC'11 | JSSC'08 | JSSC'12 | **This work** |
|---|---|---|---|---|
| Technology | 65 nm | 90 nm | 65nm | 45 nm |
| Signal Type | Any Signal | Any Signal | Any Signal | Freq.-Sparse signal |
| Size | $2^{10}$ | $2^{8}$ | $2^{7}$ to $2^{11}$ | $3^{6}x2^{10}$ |
| Word width | 16 bits | 10 bits | 12 bits | 12 bits |
| Area | 8.29mm$^2$ | 5.1mm$^2$ | 1.37mm$^2$ | 0.6mm$^2$ |
| Effective Throughput | 240MS/s | 2.4GS/s | 1.25-20MS/s | 36-109GS/s |
| Energy/Sample | 17.2pJ | 50pJ | 19.5-50.6pJ | 0.4-1.6pJ |

# Conclusion

- 0.75 million-point sparse FFT in 45nm CMOS IBM SOI
- More than 40x better energy efficiency than traditional FFTs
- 88x improvement in run-time over the software implementation

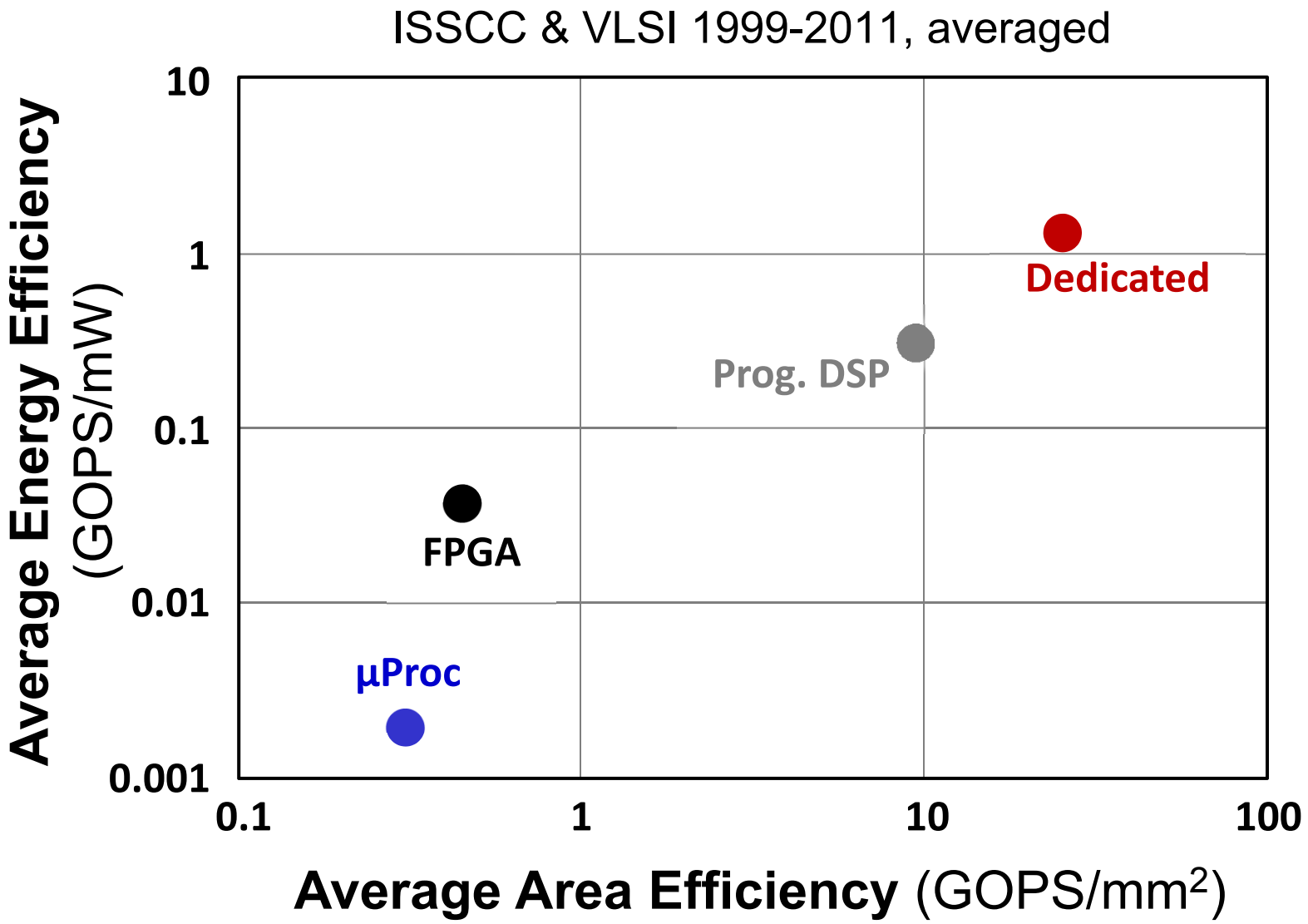
# Paper 27.5

# A Multi-Granularity FPGA with Hierarchical Interconnects for Efficient and Flexible Mobile Computing

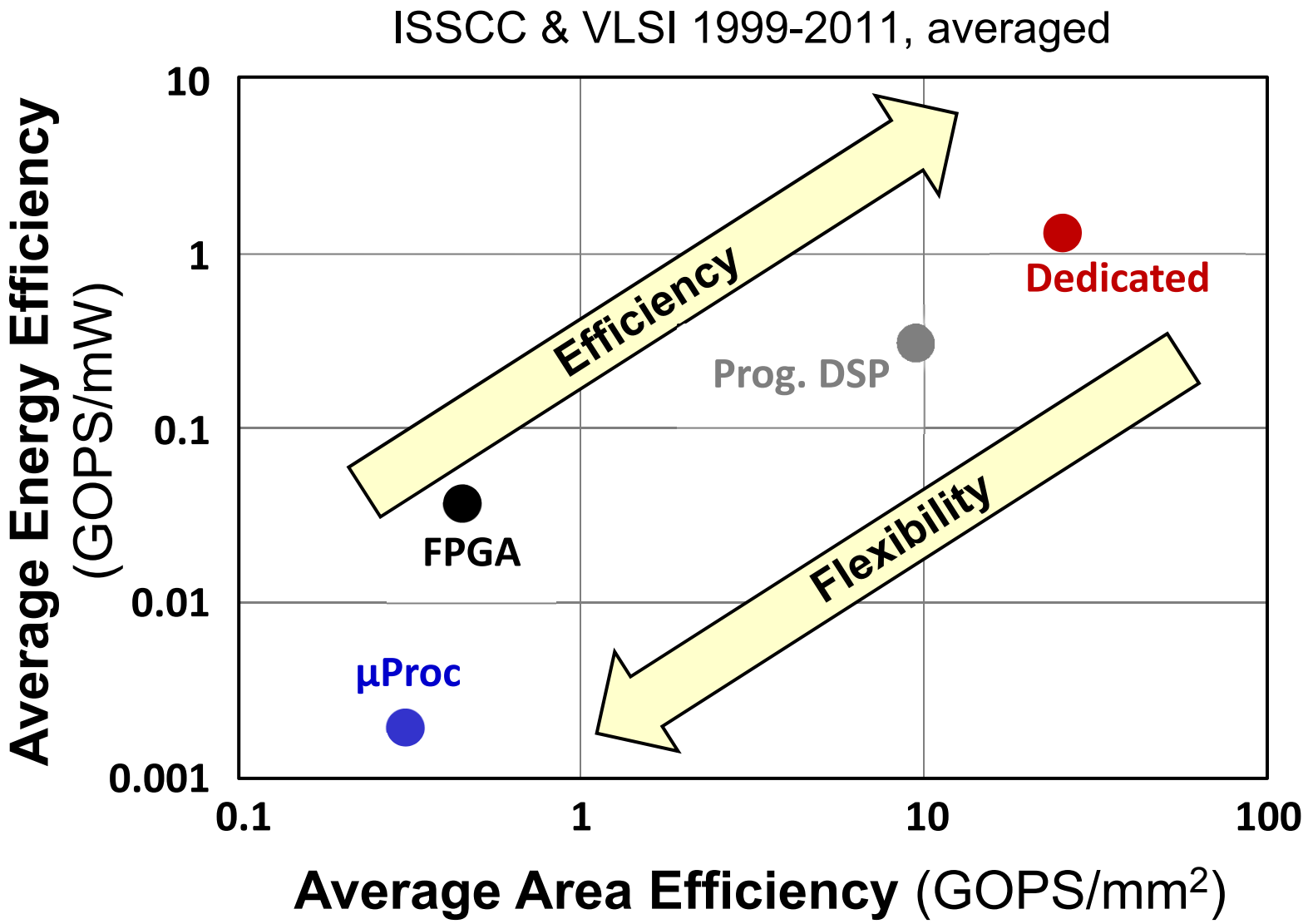
Cheng C. Wang[1], Fang-Li Yuan[1],
Tsung-Han Yu[2], Dejan Marković[1]

[1]University of California, Los Angeles, CA
[2]Qualcomm, Irvine, CA

# How Efficient are Today's Chips?

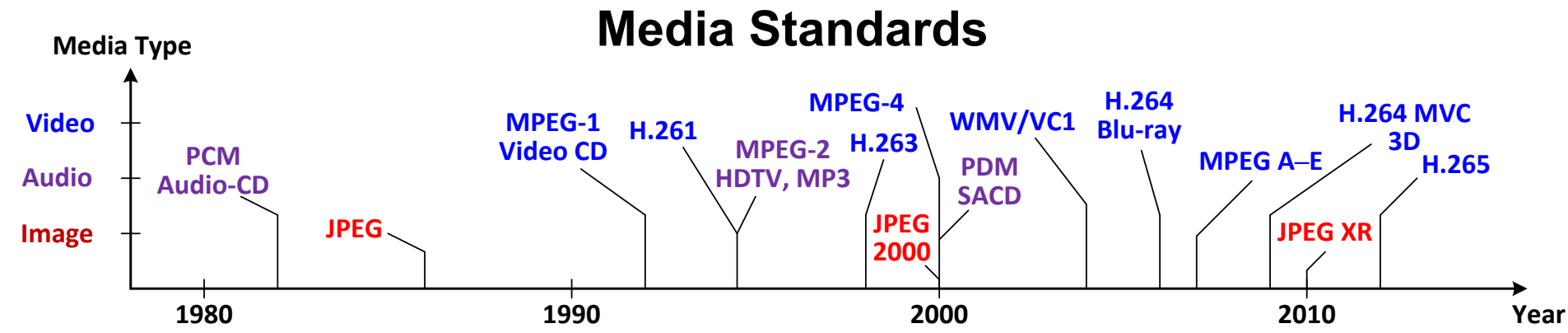


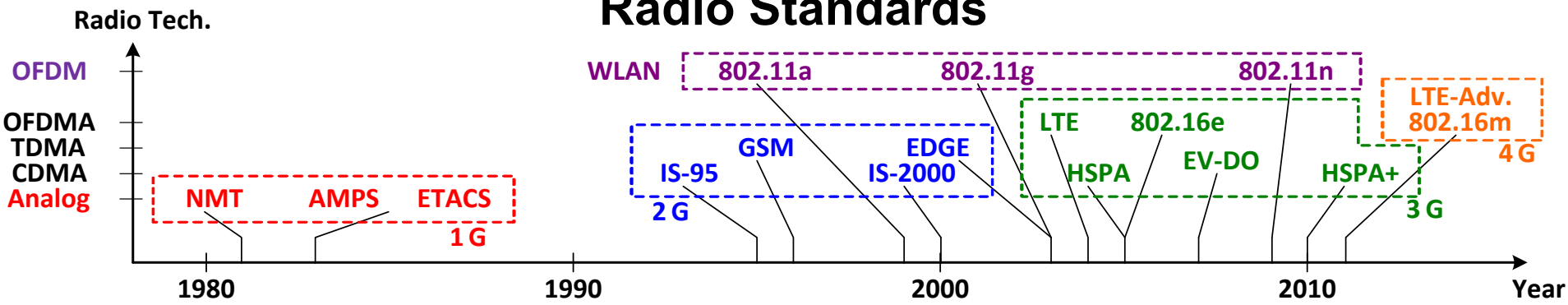
# Efficiency vs. Flexibility



# Issue 1: Design and Re-design

- New standards + new features = frequent re-design
- Not a scalable solution under rising design costs

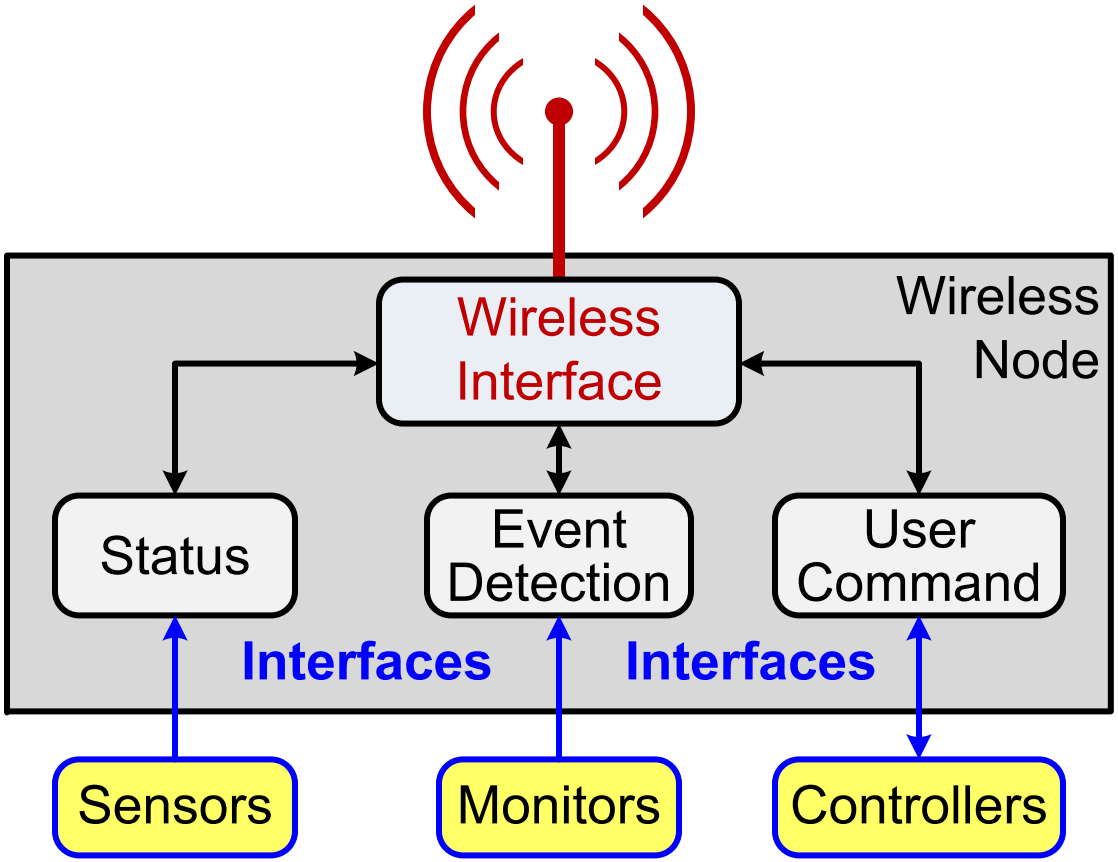


# Issue 2: Non-standard Interfaces

**Growing # of connected devices (e.g. Internet of Things)**

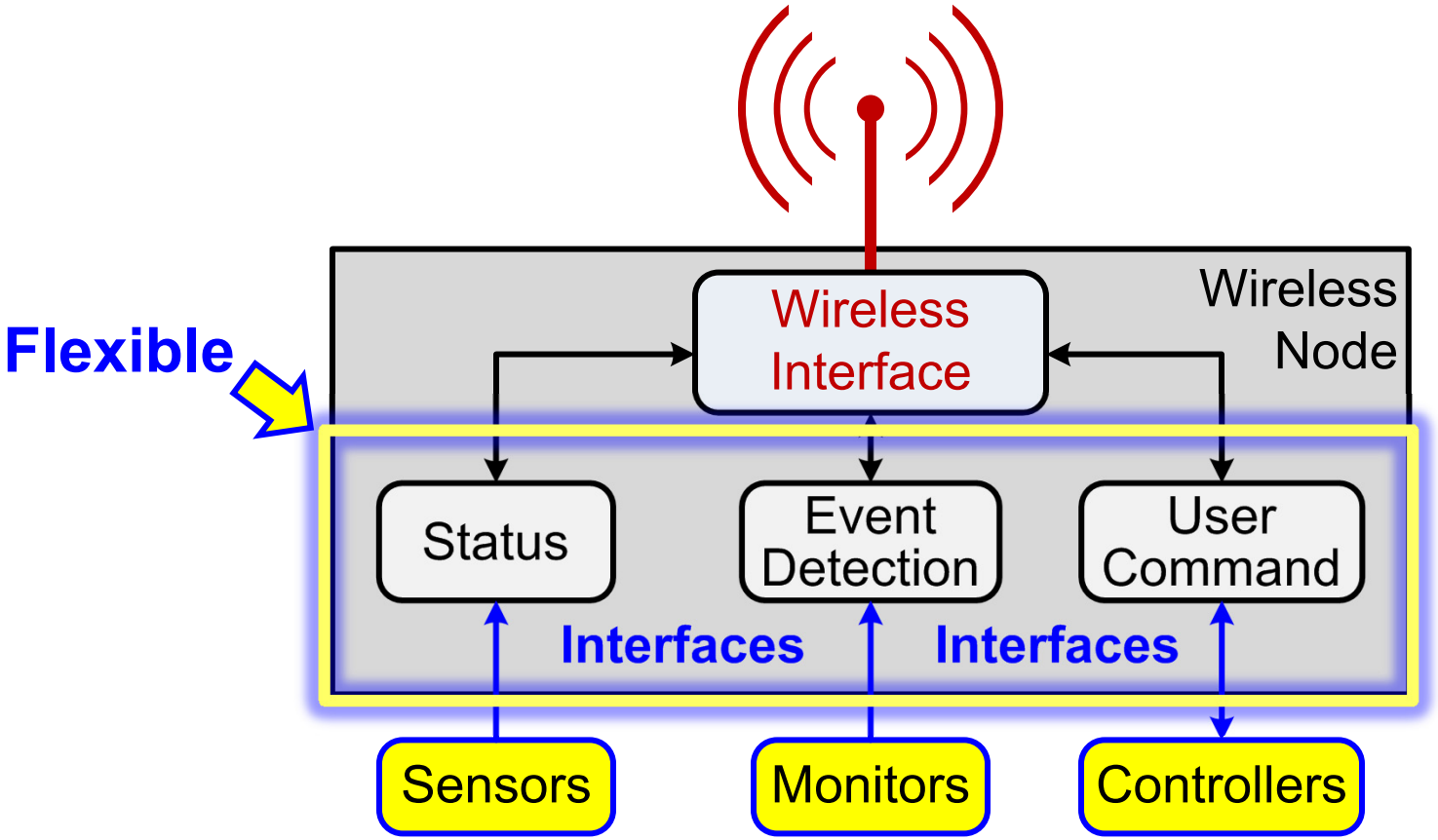
- ✖ Many devices = many interfaces
- ✖ Most interfaces are non-standard



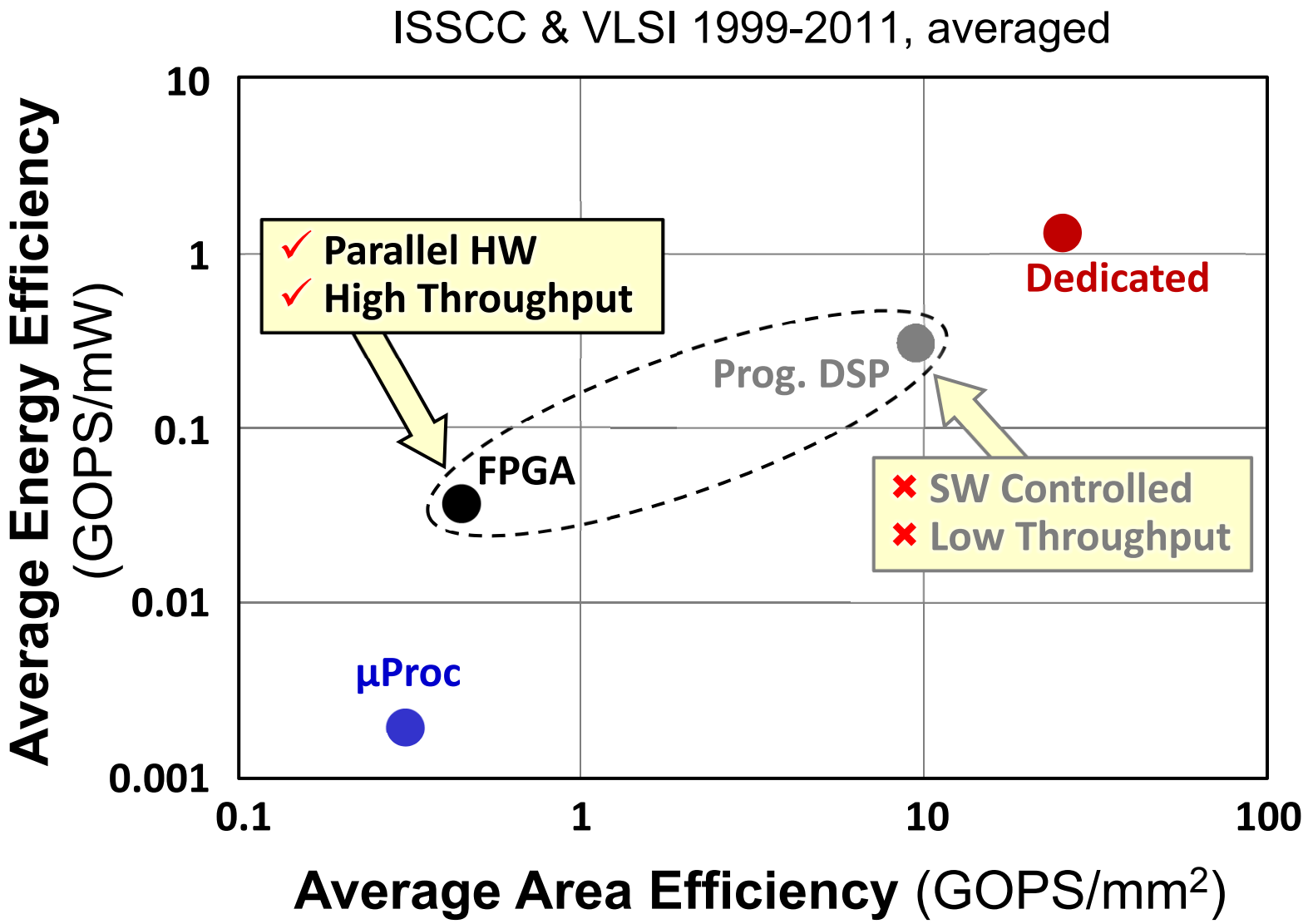
# Solution: Flexible IO, Flexible HW

**Growing # of connected devices (e.g. Internet of Things)**

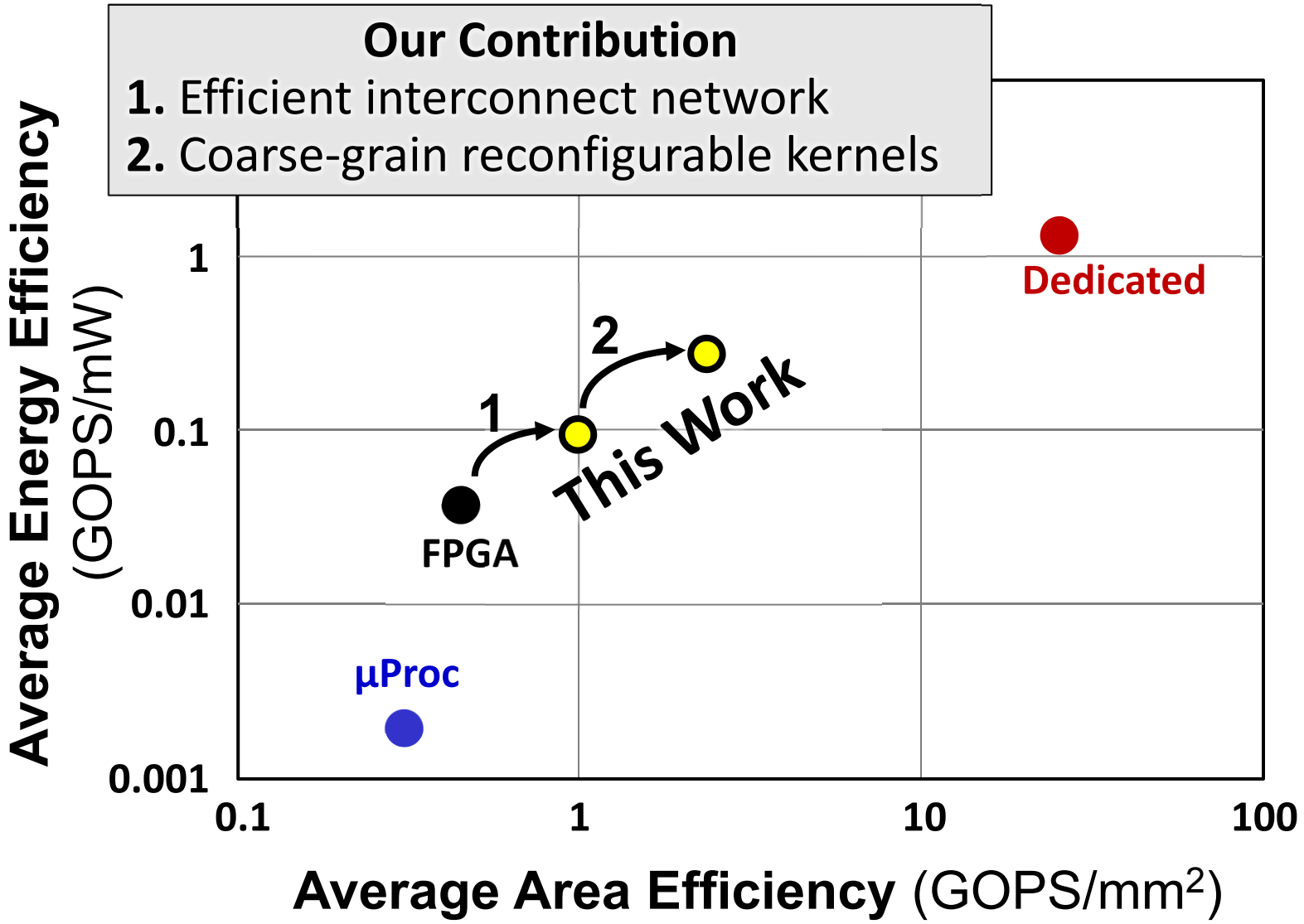
- ✖ Many devices = many interfaces
- ✖ Most interfaces are non-standard



# Candidates for Flexibility



# Two Steps Towards Higher Efficiency



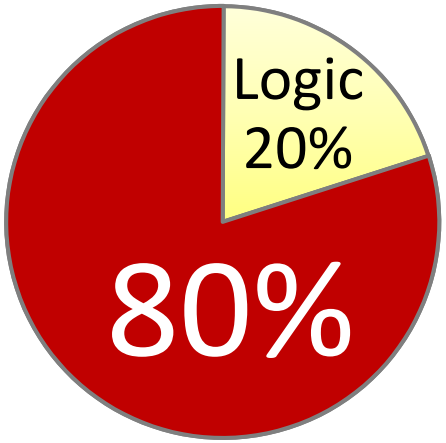
# Why are FPGAs Inefficient?

- **Compared to ASICs, FPGAs incur penalties in** [1]
  - Area ( 17 – 70 x )
  - Speed ( 3 – 6 x )
  - Power ( 5 – 52 x )

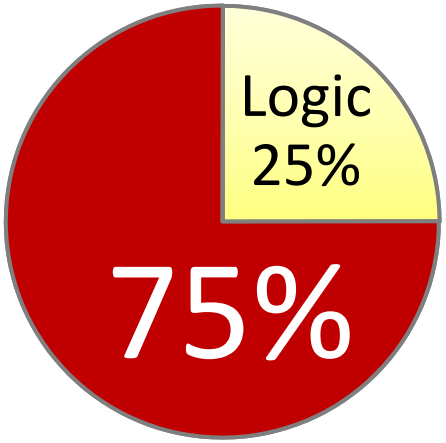
× **Area** Efficiency gap

× **Energy** Efficiency gap

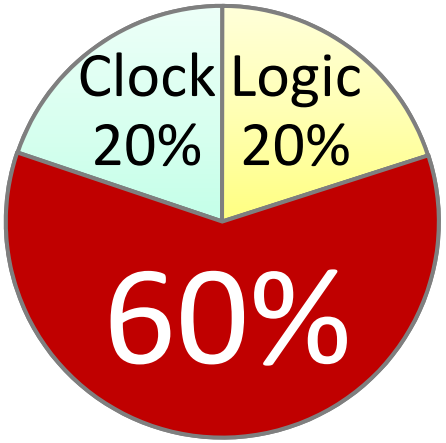
*Area* [2]




*Delay*



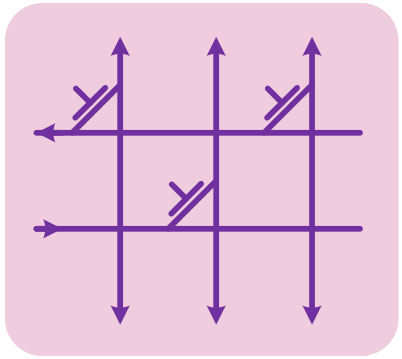

*Power*



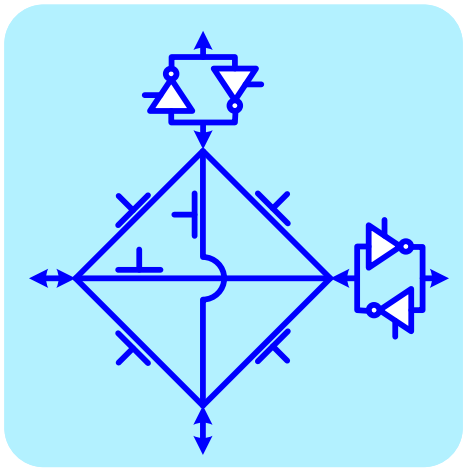

**Interconnect**

[1] I. Kuon & J. Rose, *IEEE TCAD-ICS*, Feb. 2007
[2] I. Bolsens, *MPSOC* 2006

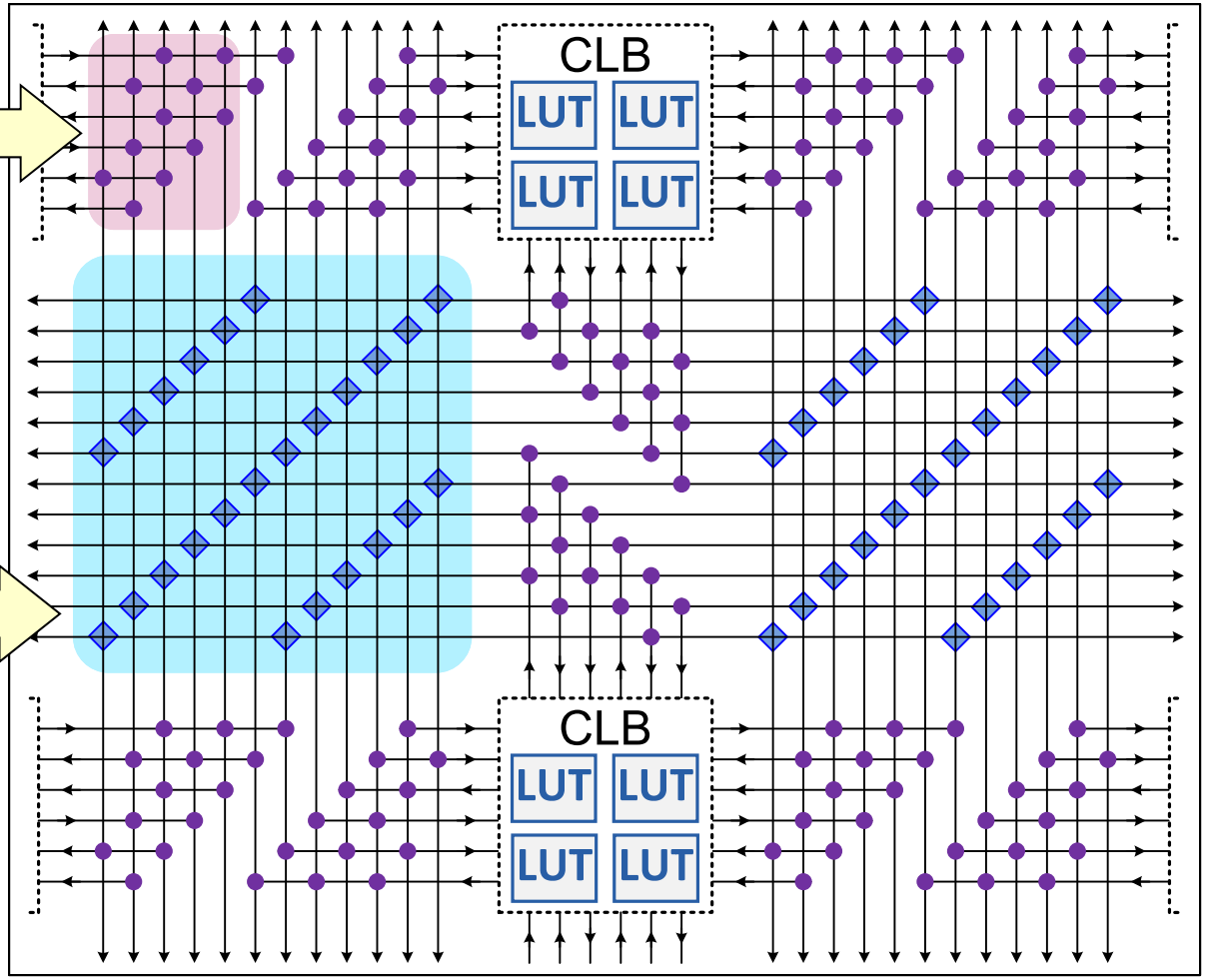
# Current Architecture is Not Scalable



**O($N^2$)** worst-case complexity



## 2D-Mesh Interconnects



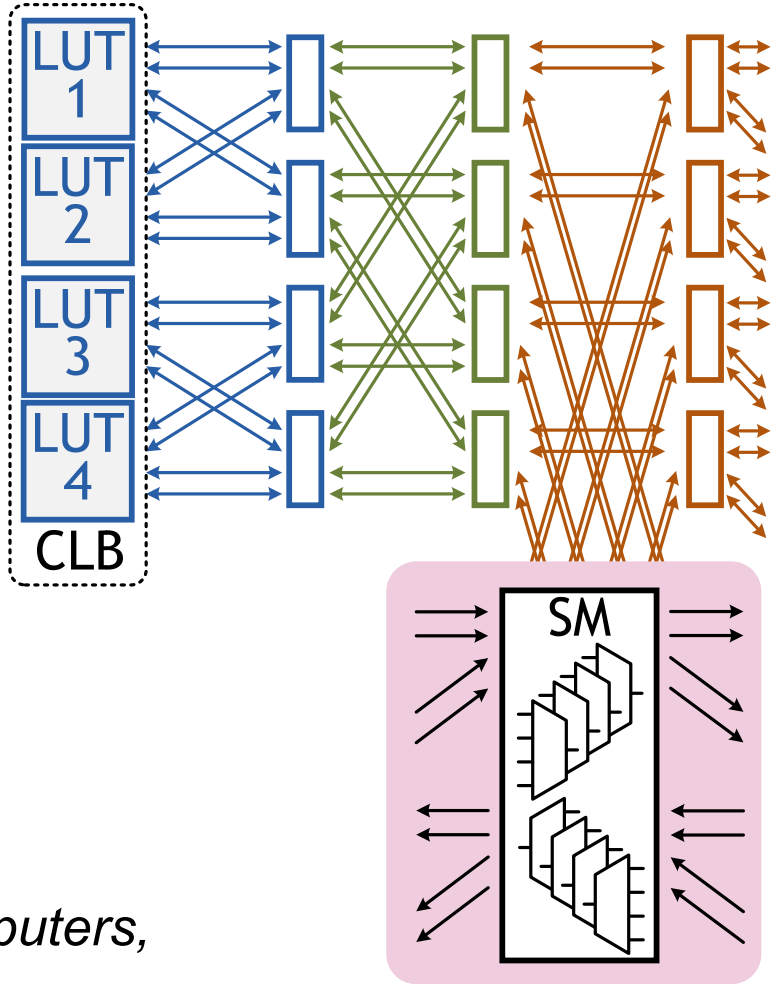
# Hierarchical Interconnect Architecture

Unidirectional routing using Switch Matrix (SM)

Full connectivity

**O(*N*·log*N*)** complexity

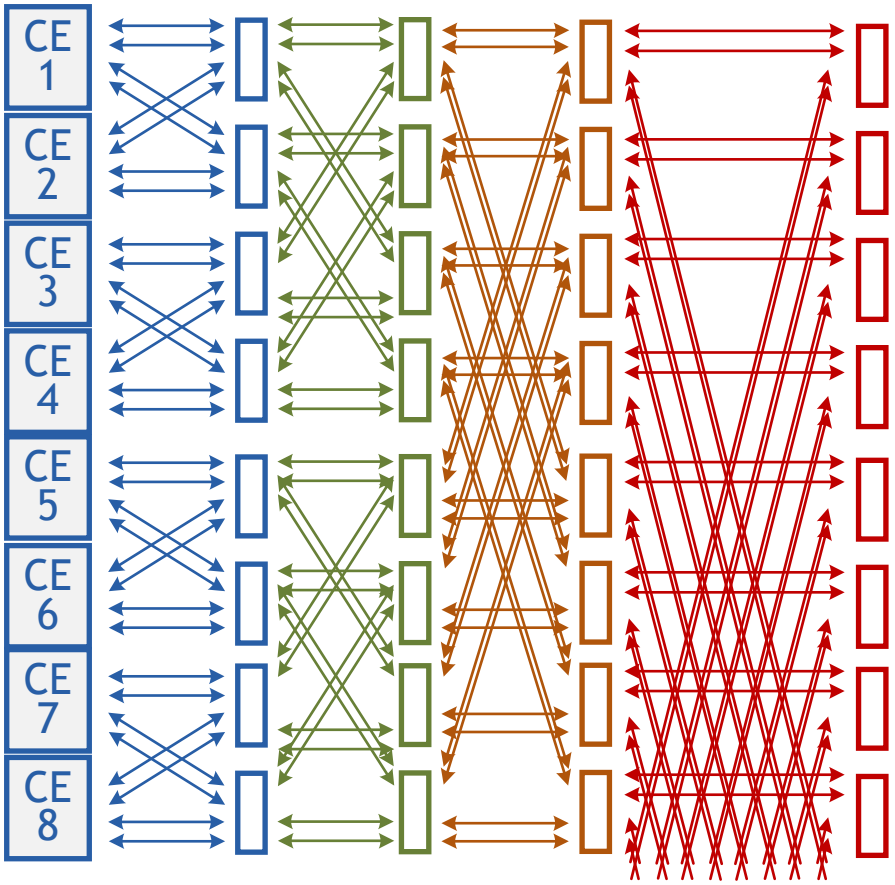
**Folded Beneš Network** [3]




[3] C.E. Leiseron, *IEEE Trans. Computers, Oct. 1985*.

# Traditional Radix-2 Beneš Network

- **Locality problem at radix boundaries**
  - First perform an isomorphic transformation [4]

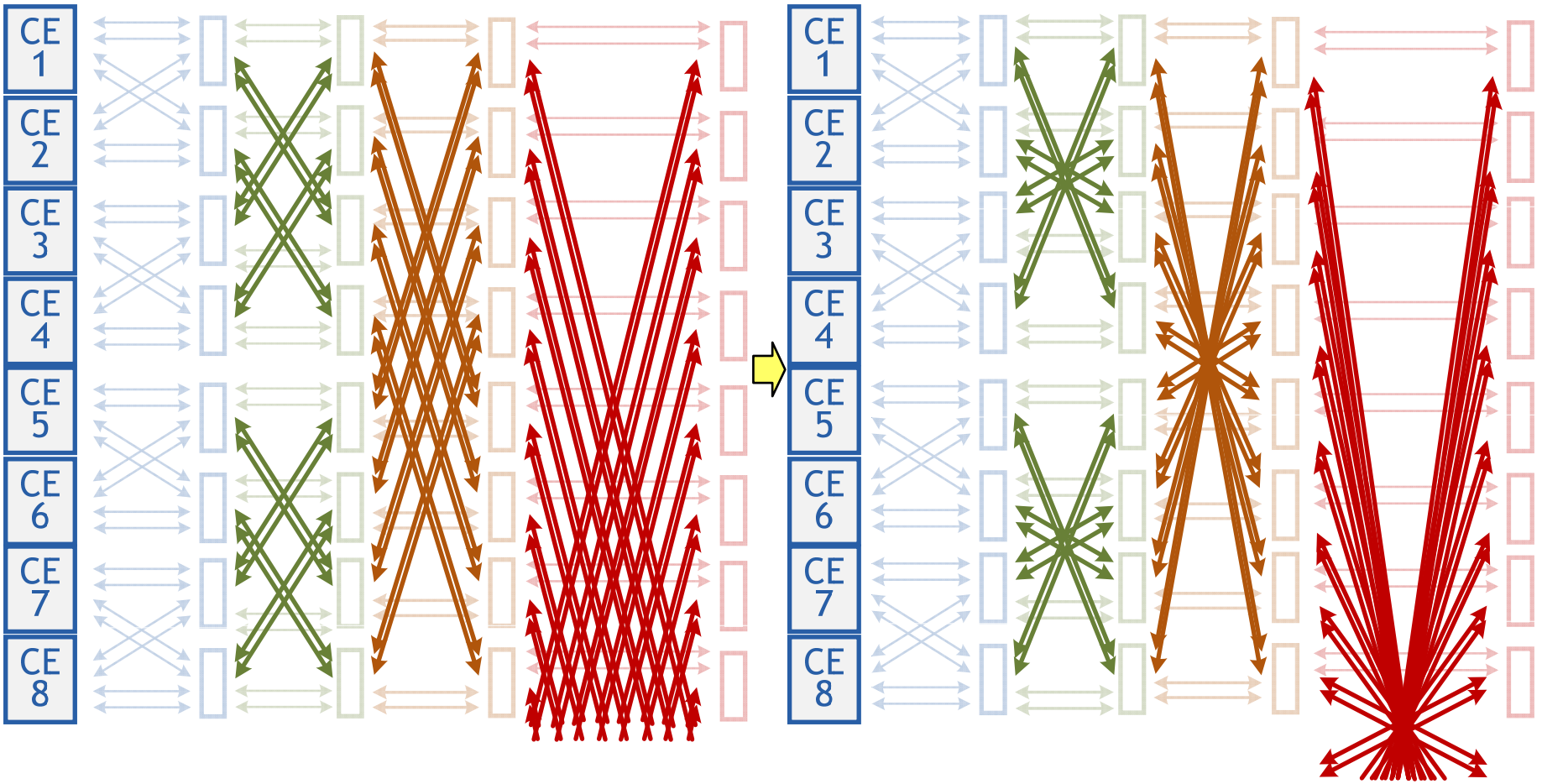



[4] C.-L. Wu, Trans. on Computers, 1980

# Transformed Isomorphic Network

## Shorter cross-routes across bi-sections

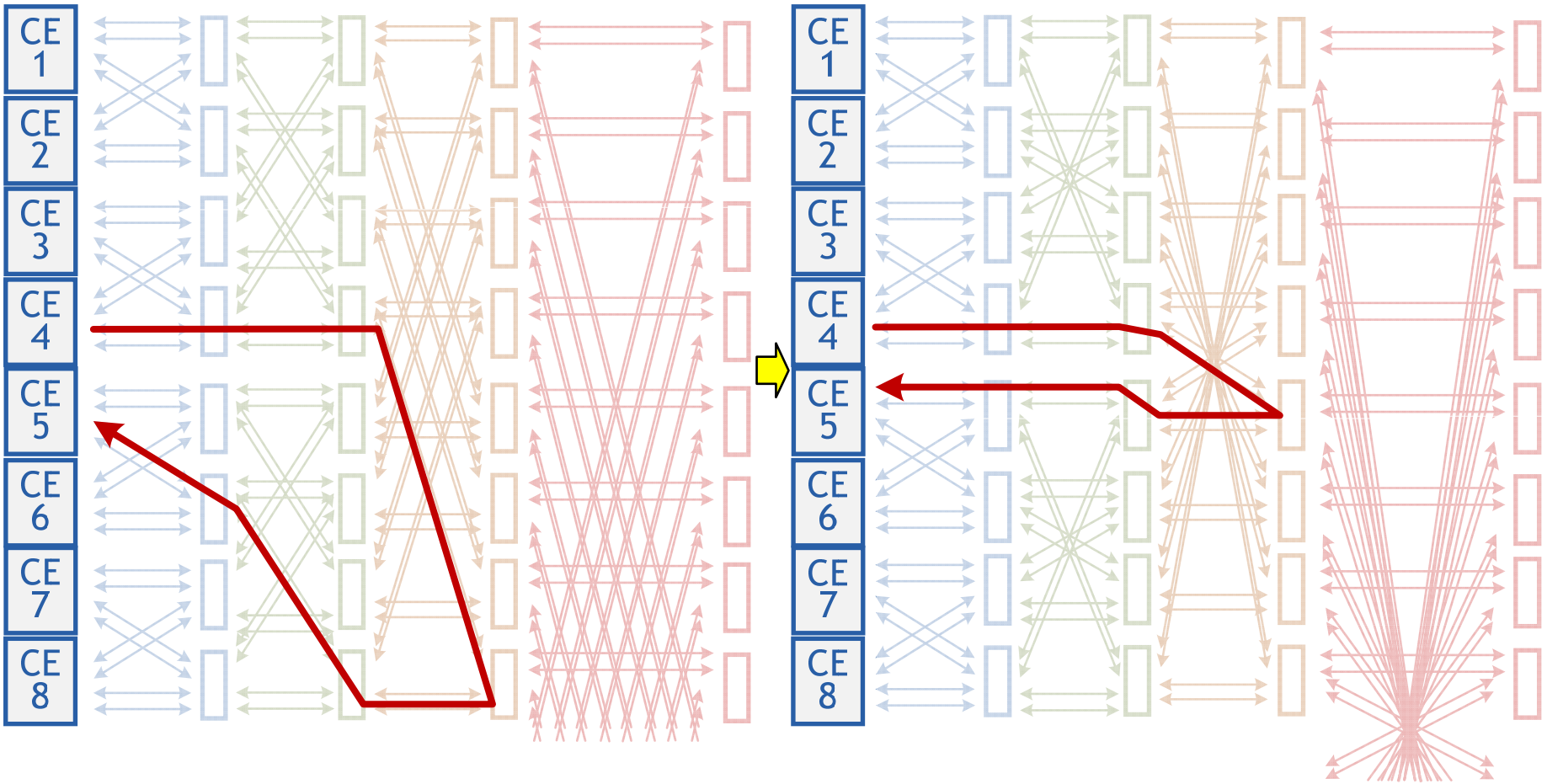
*Original* *Transformed*



# But… Radix Boundaries Still Exist

## Shorter path, but still 3 hierarchies

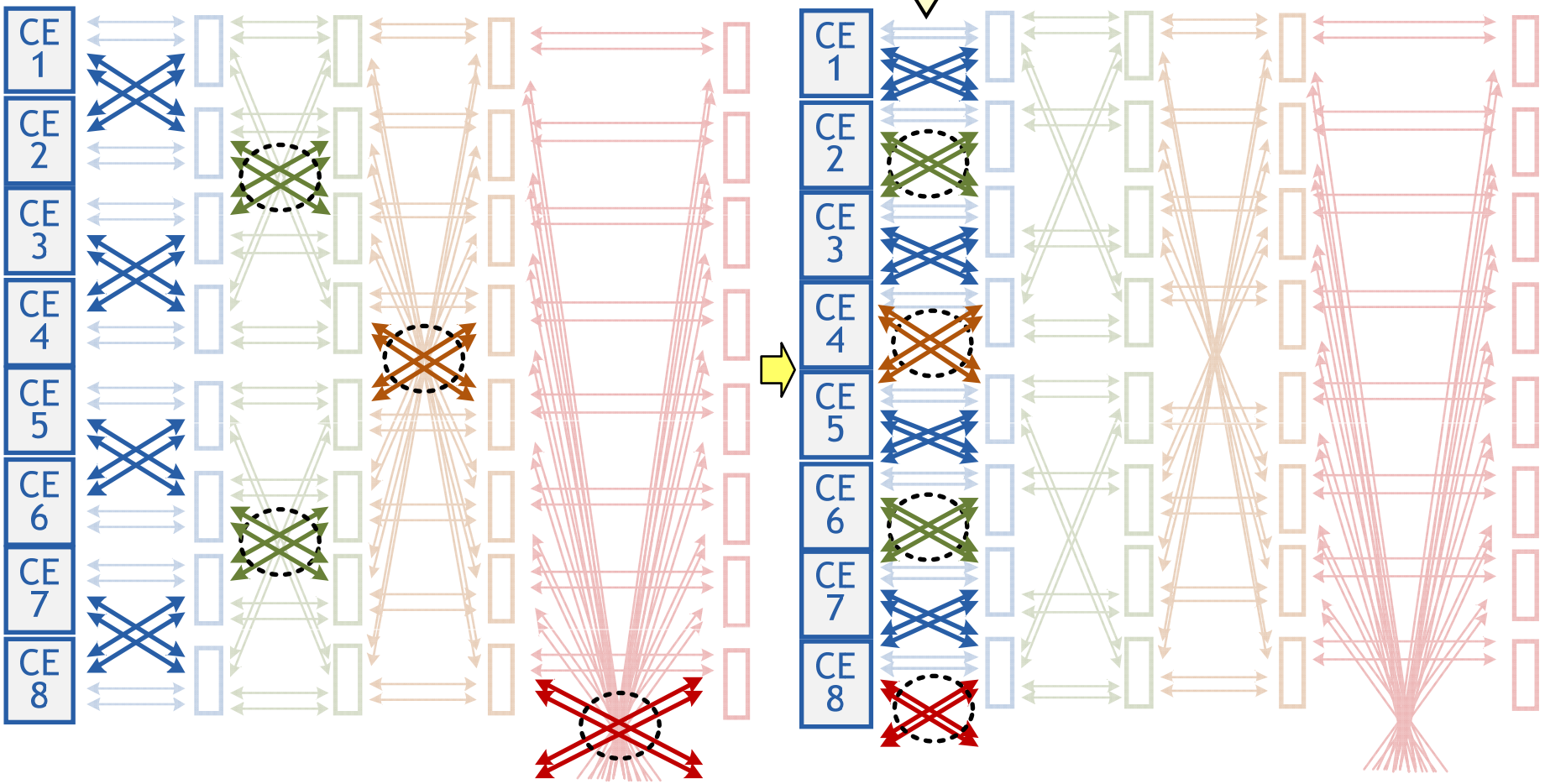
*Original* *Transformed*



# Radix-3 Boundary-less Interconnect

*Move circled nets down to the **1st stage***

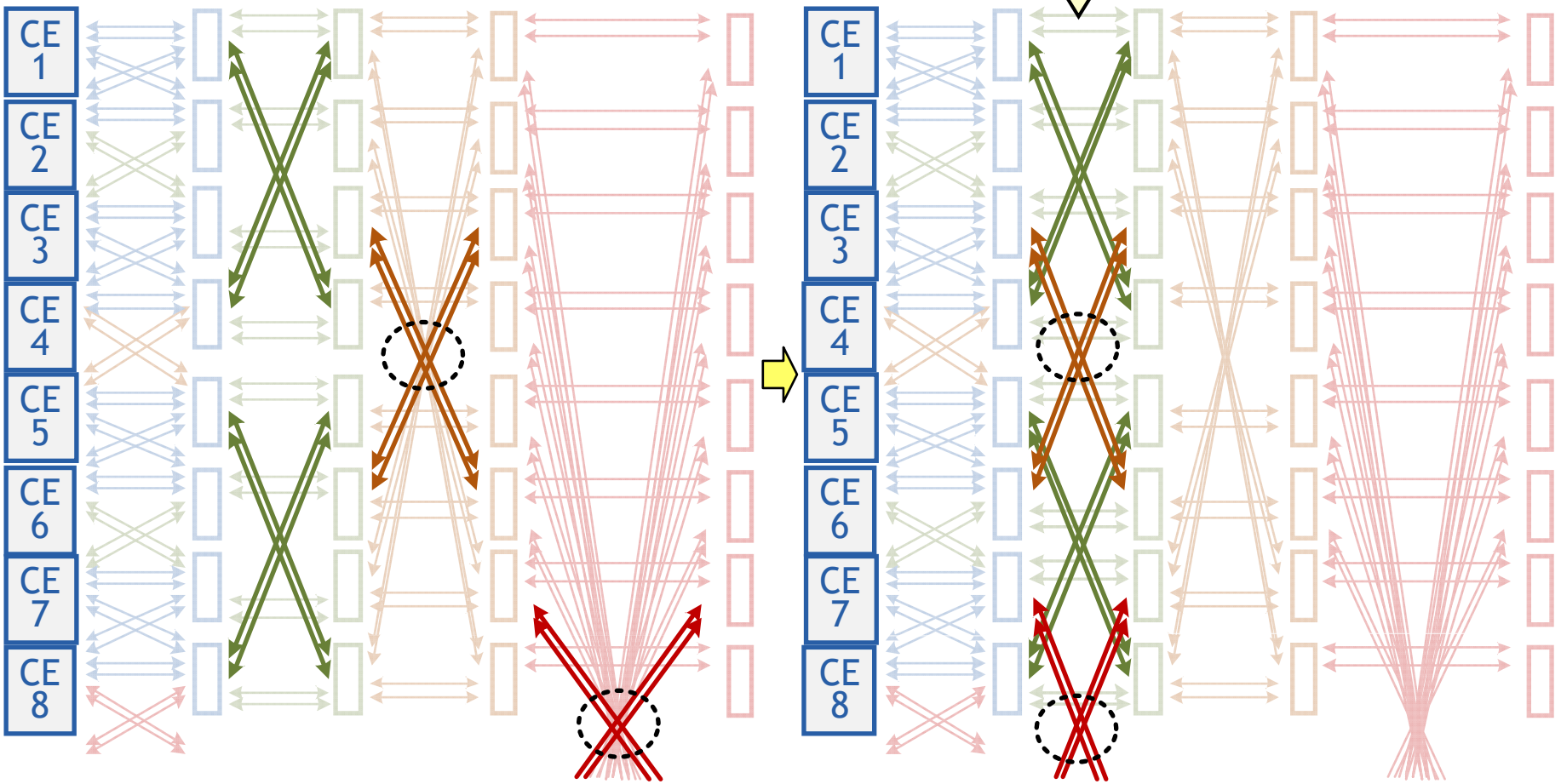
*Restored **spatial locality** at the **1st hierarchy level***



# Continue Recursively: 2nd Stage

*Move circled nets down to the **2nd stage***

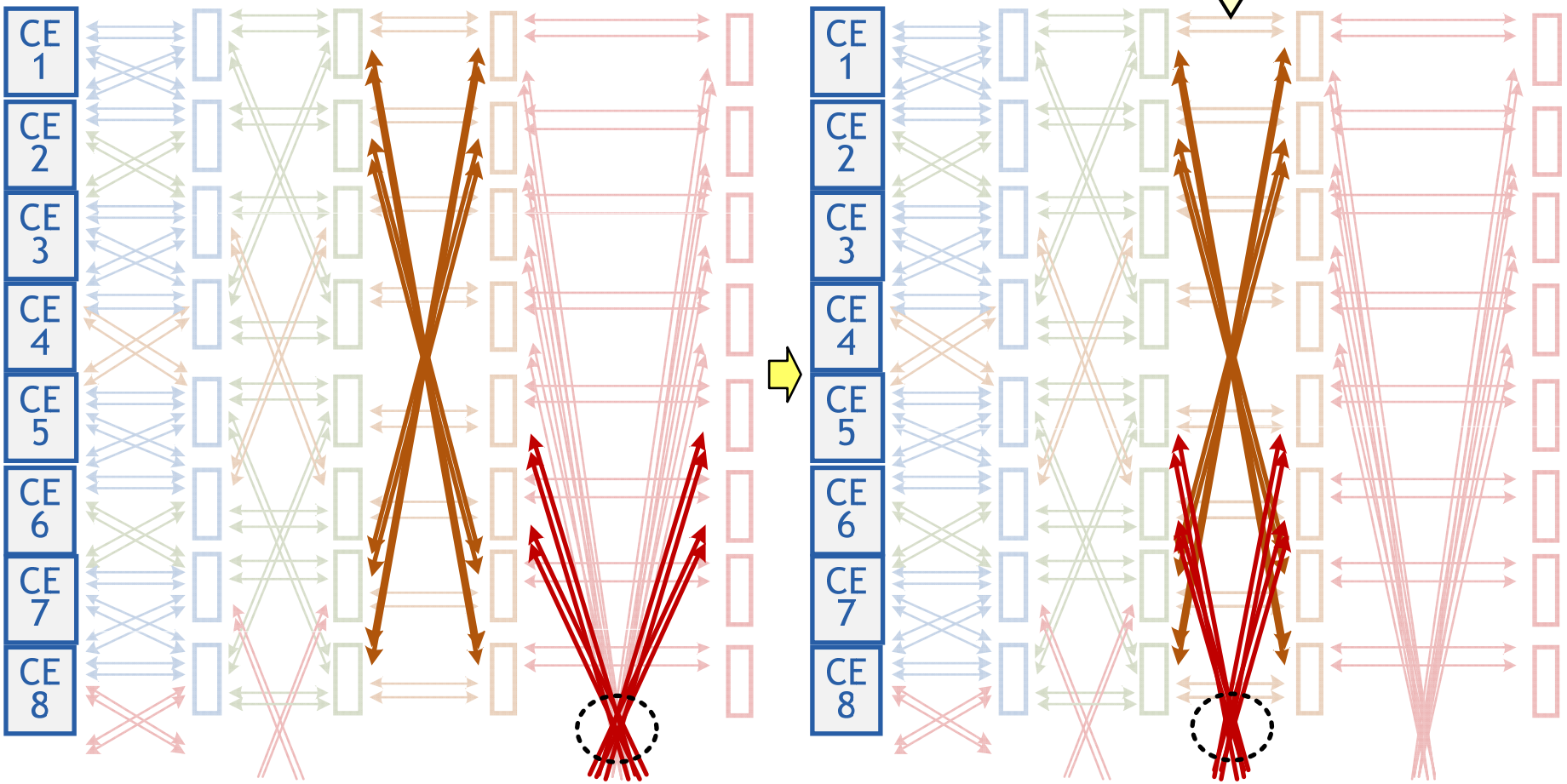
*Restored **spatial locality** at the **2nd hierarchy level***



# Continue Until Desired Hierarchy

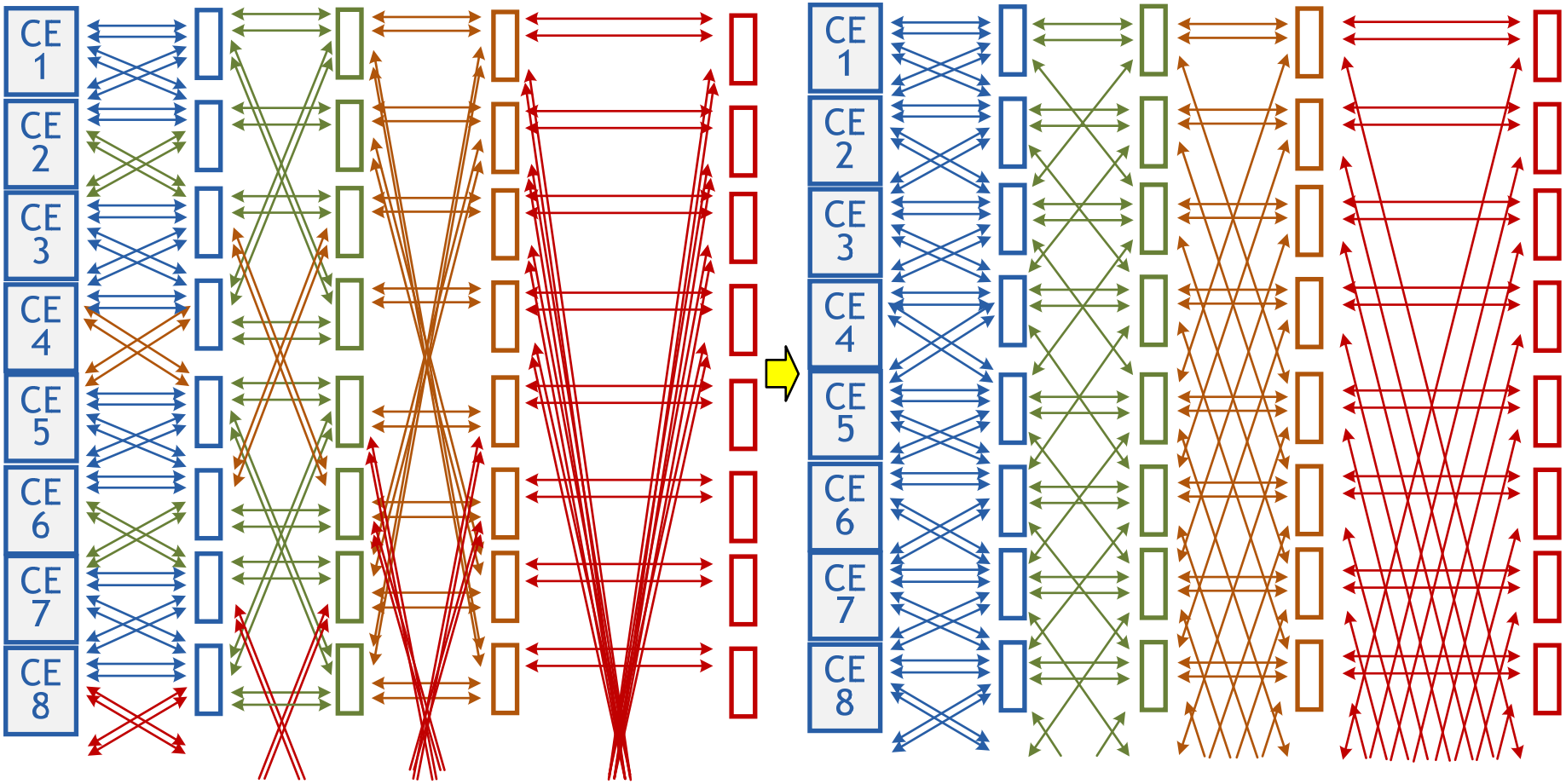
*Move circled nets down to the **3rd stage***

*Restored **spatial locality** at the **3rd hierarchy level***



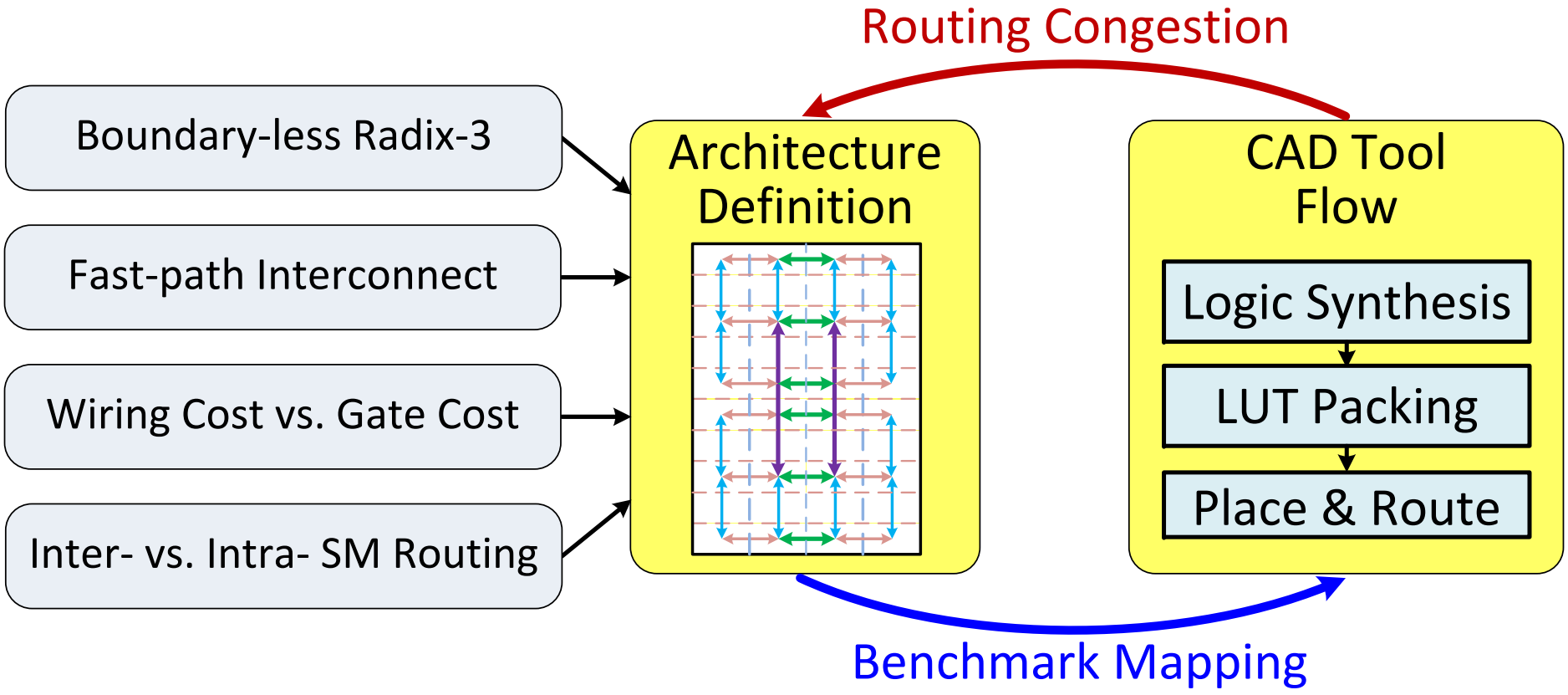
# Final Step: Re-Distribute Wires

## Wiring uniformity simplifies integration



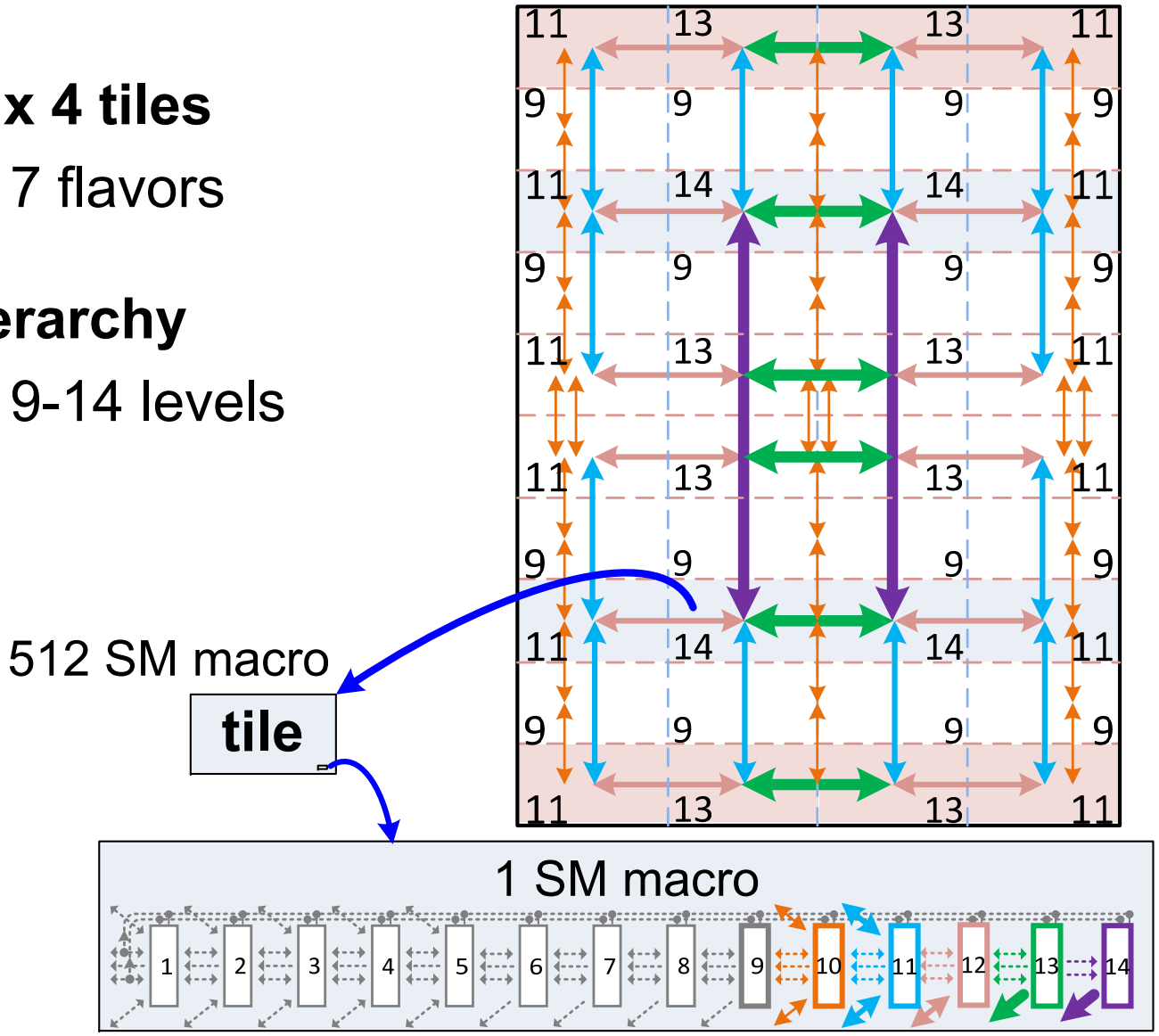
# Closed-Loop Architecture Design

- **Design a candidate architecture**
- **Map benchmark designs**
- **Check routing congestion and timing**



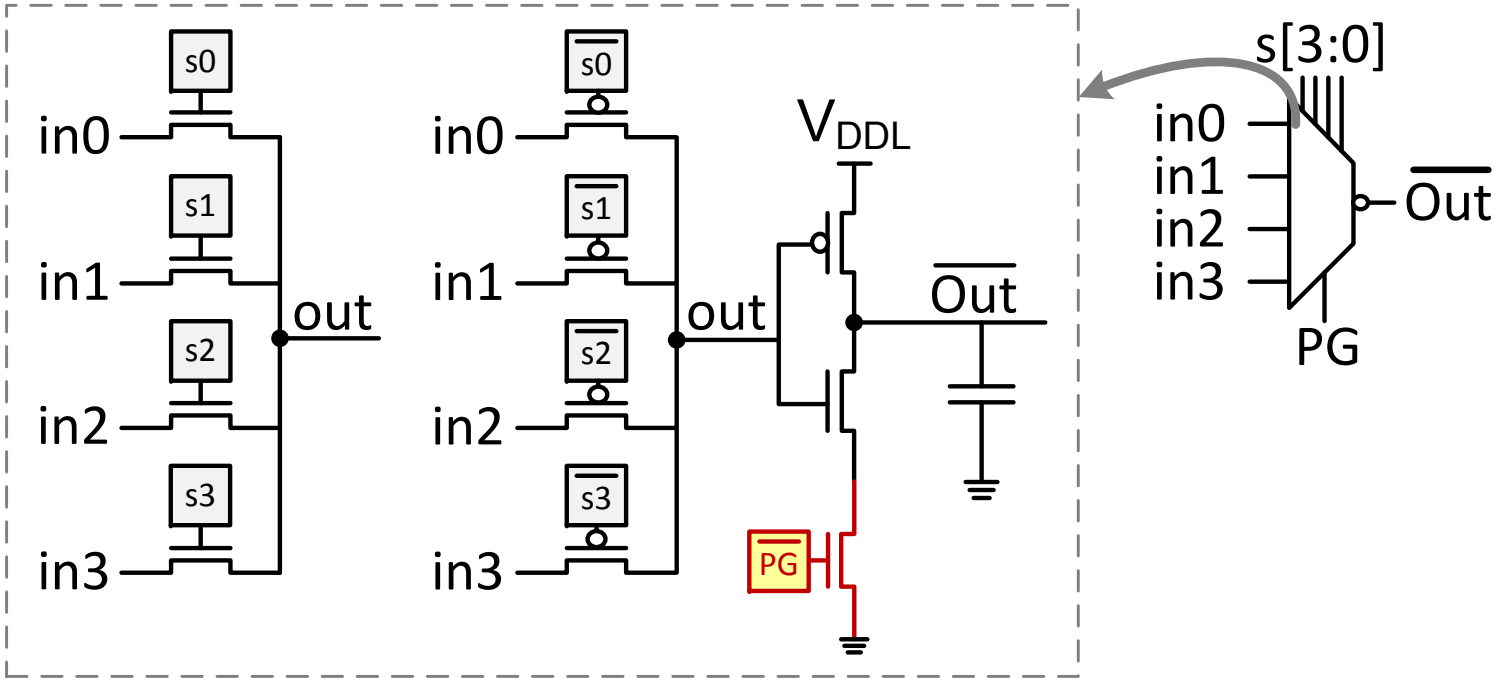
# Top-level Interconnect Architecture

- **10 x 4 tiles**
  - 7 flavors
- **Hierarchy**
  - 9-14 levels



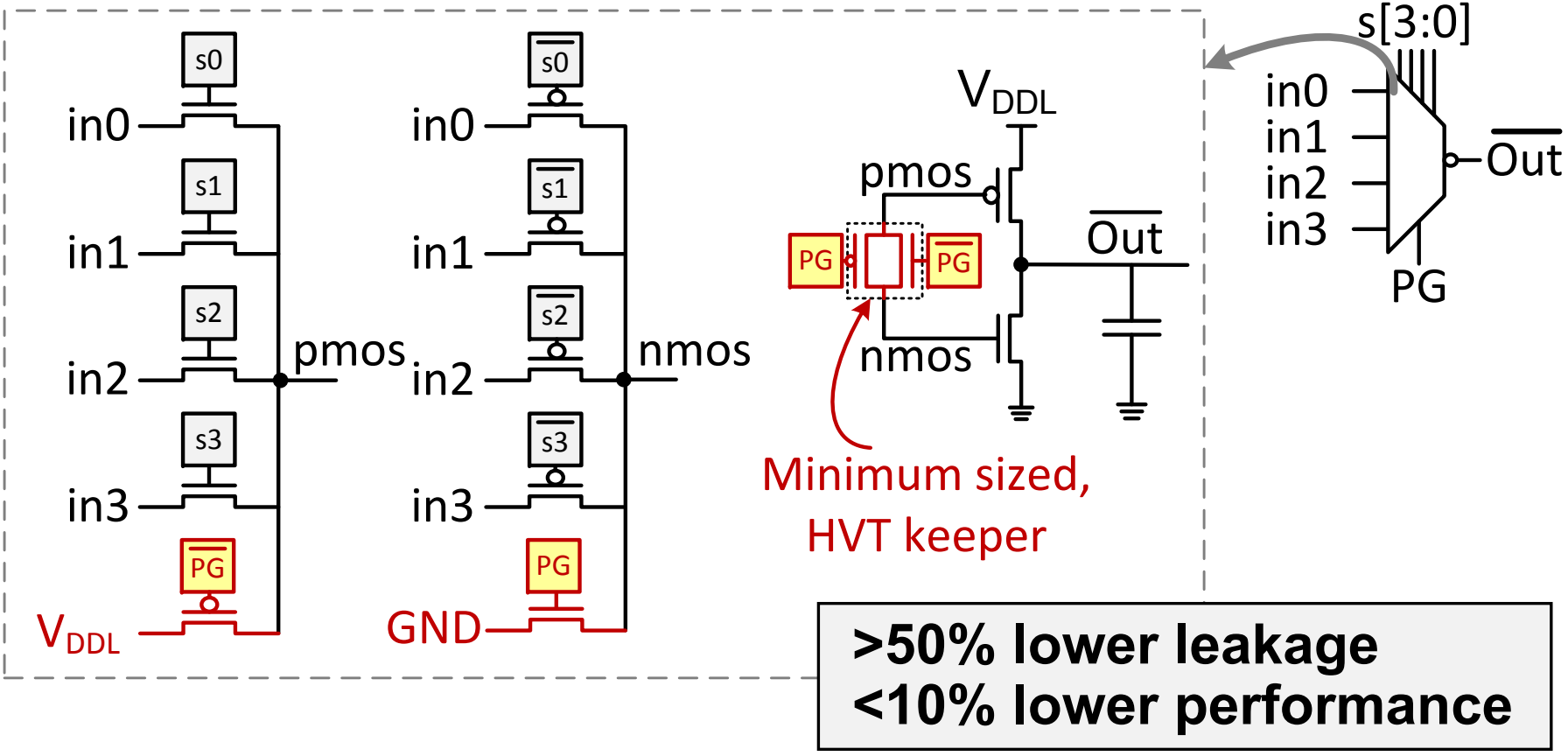
# Conventional Power Gating

- **Interconnect power dominates in today's FPGAs**
  - ❑ Block-level PG is ineffective for interconnects
  - × PG individual inverter requires large footer
  - × Large area penalty or large performance penalty



# Proposed Fine-Grain Power Gating

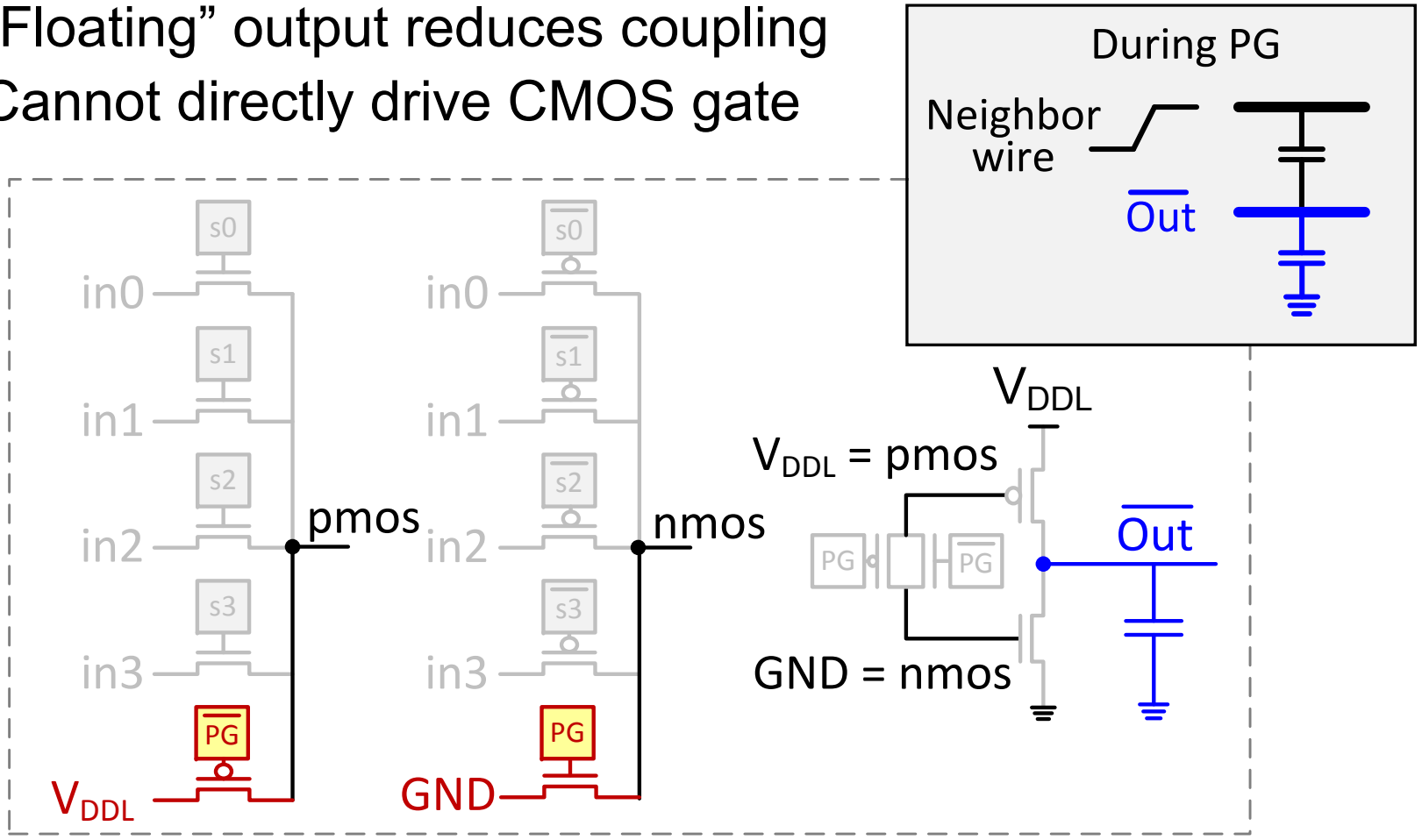
- Additional mux input for PG
- Separate NMOS and PMOS branches
- Small keeper pass-gate for swing-restoration



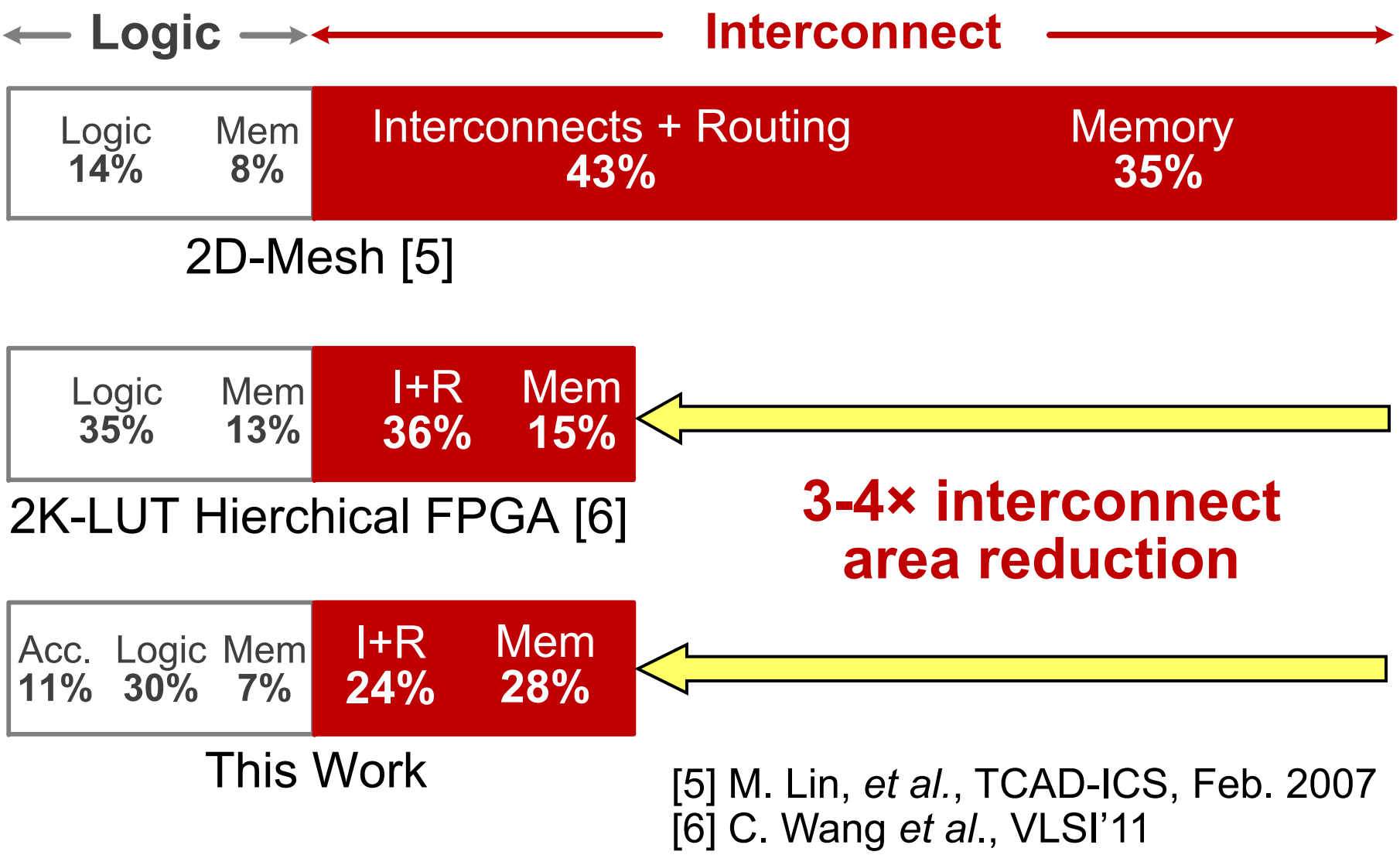

**>50% lower leakage**
**<10% lower performance**

# Power Gating Mode

- **During power gating:**
  - PMOS and NMOS of the output inverter both turn off
  - "Floating" output reduces coupling
  - × Cannot directly drive CMOS gate



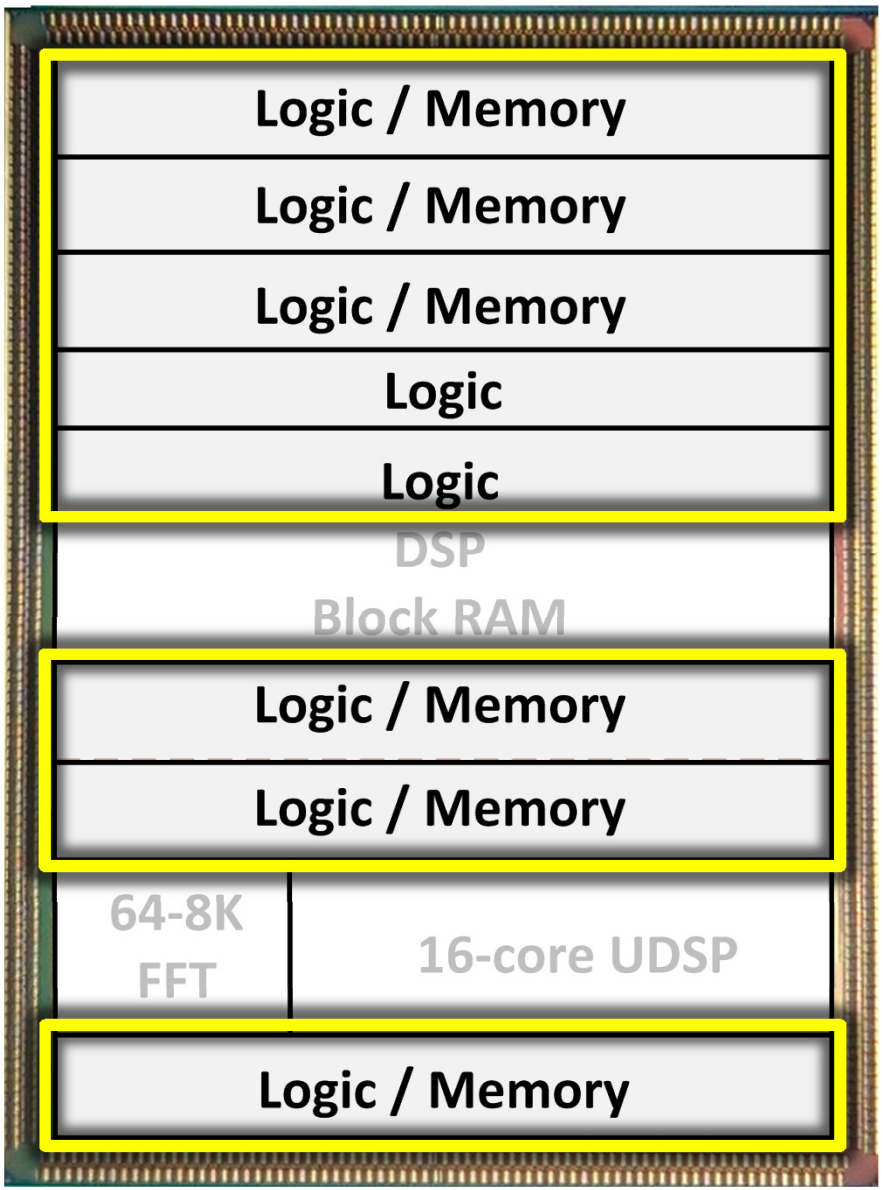
# Logic : Interconnect Area = 1 : 1





[5] M. Lin, *et al.*, TCAD-ICS, Feb. 2007
[6] C. Wang *et al.*, VLSI'11

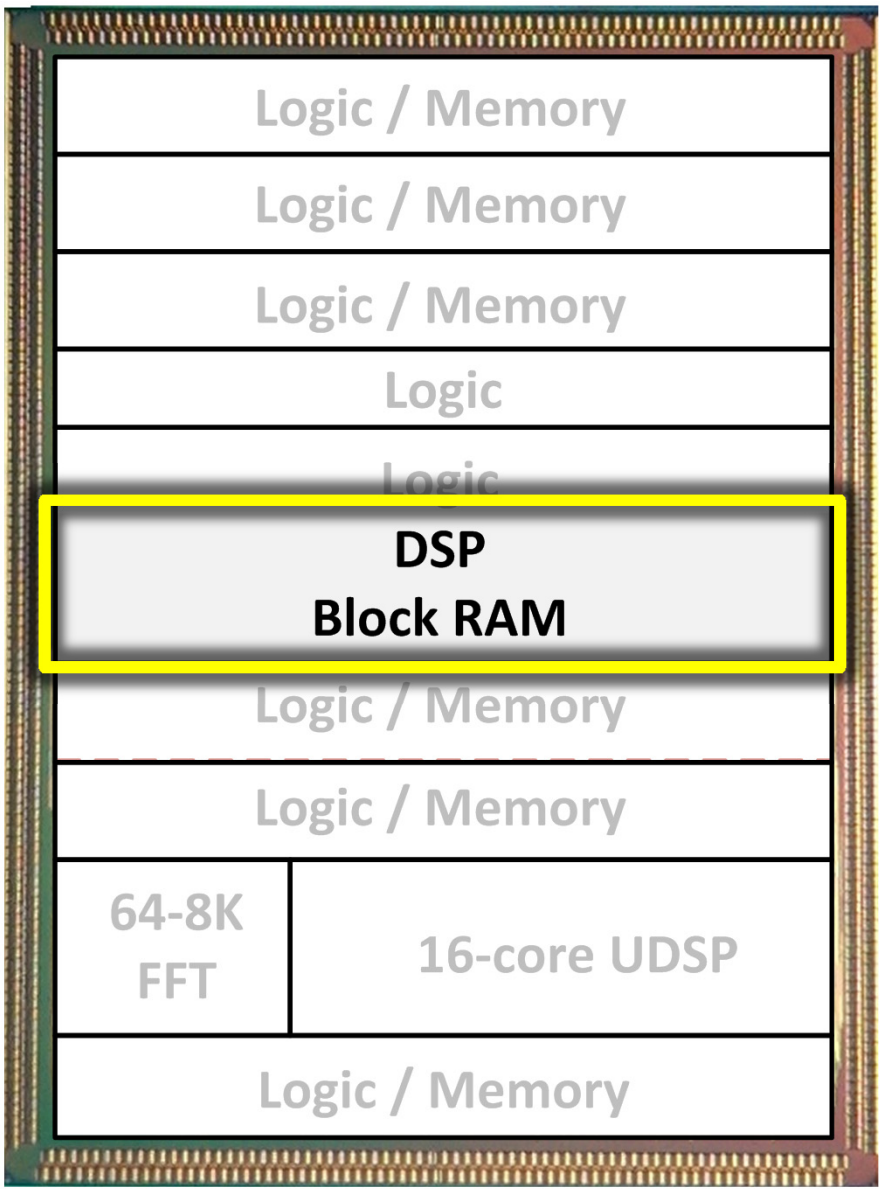
# Heterogeneous Logic – Fine Grain




**3 granularities of configurable logic blocks (CLBs)**

- **Fine Grain**
  - 2752 CLBs (11K 6-in LUTs)
  - 576 CLBs with RAM and shift register
  - Virtex-6/7 compatible
  - 350 data I/Os (171 bonded)

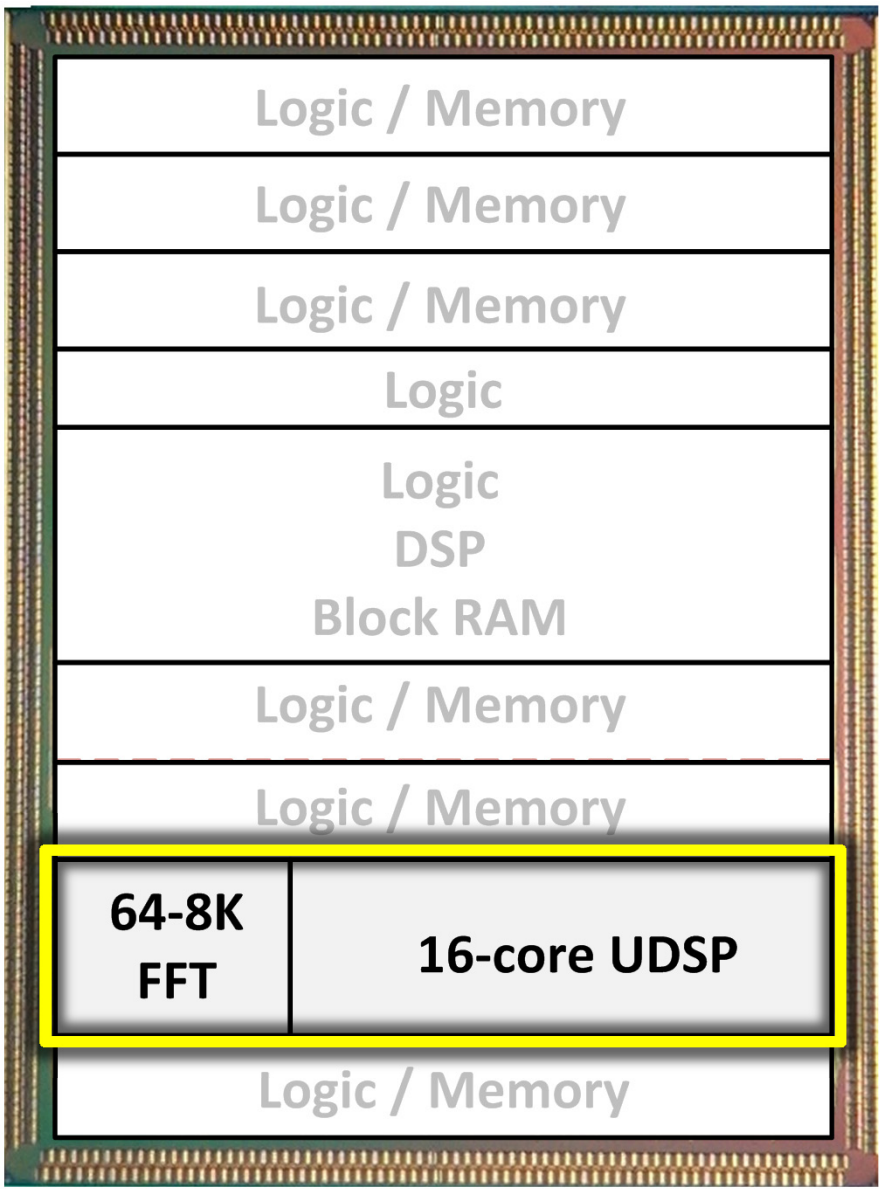
# Heterogeneous Logic – Medium Grain




**3 granularities of configurable logic blocks (CLBs)**

- **Medium Grain**
  - 42 DSP accelerators
  - 48-bit MAC and SIMD
  - 16 dual-port Block RAM (BRAM)
  - Reconfigurable from 36K×1b to 512×72b
  - Virtex-6/7 compatible

# Heterogeneous Logic – Coarse Grain

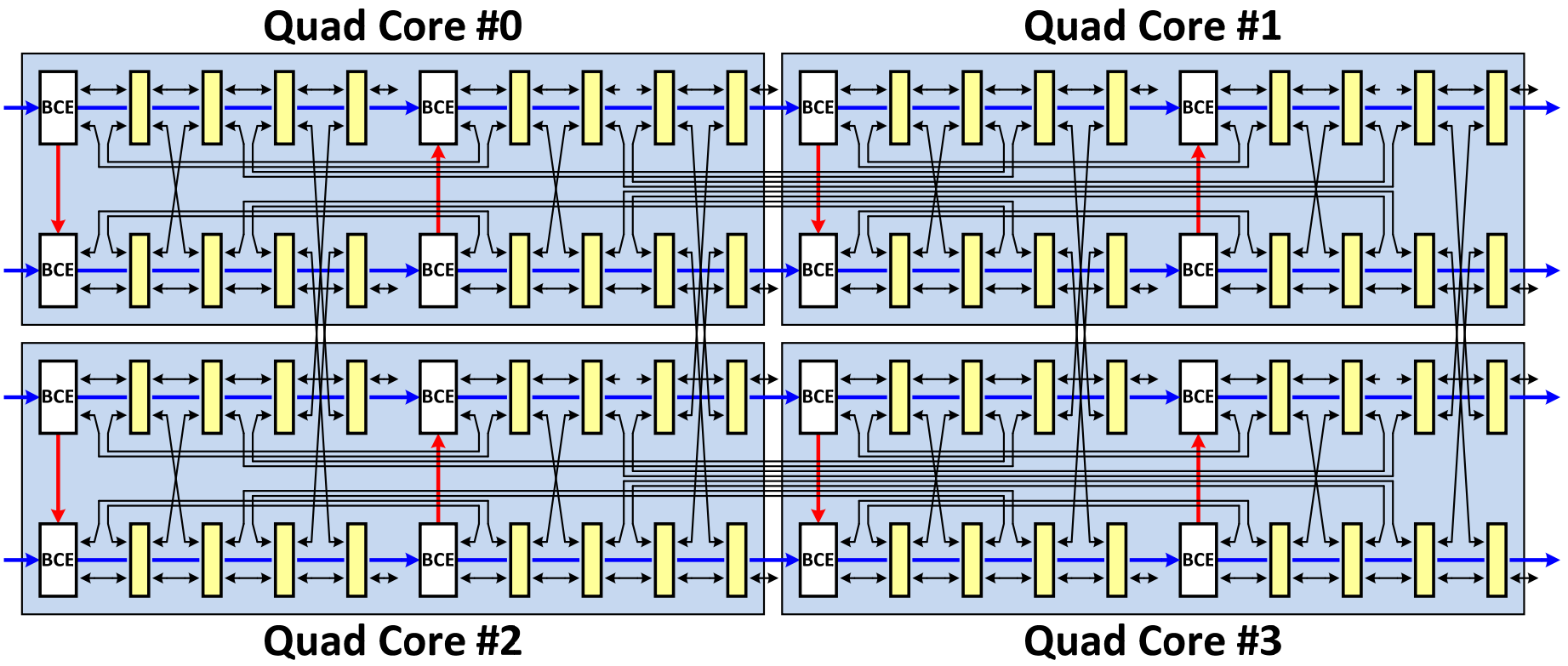



**3 granularities of configurable logic blocks (CLBs)**

- **Coarse Grain**
  - 16-core universal DSP
  - Targets communication and software-defined radio applications
  - 16-core reconfigurable FFT
  - 64-8192 point FFT

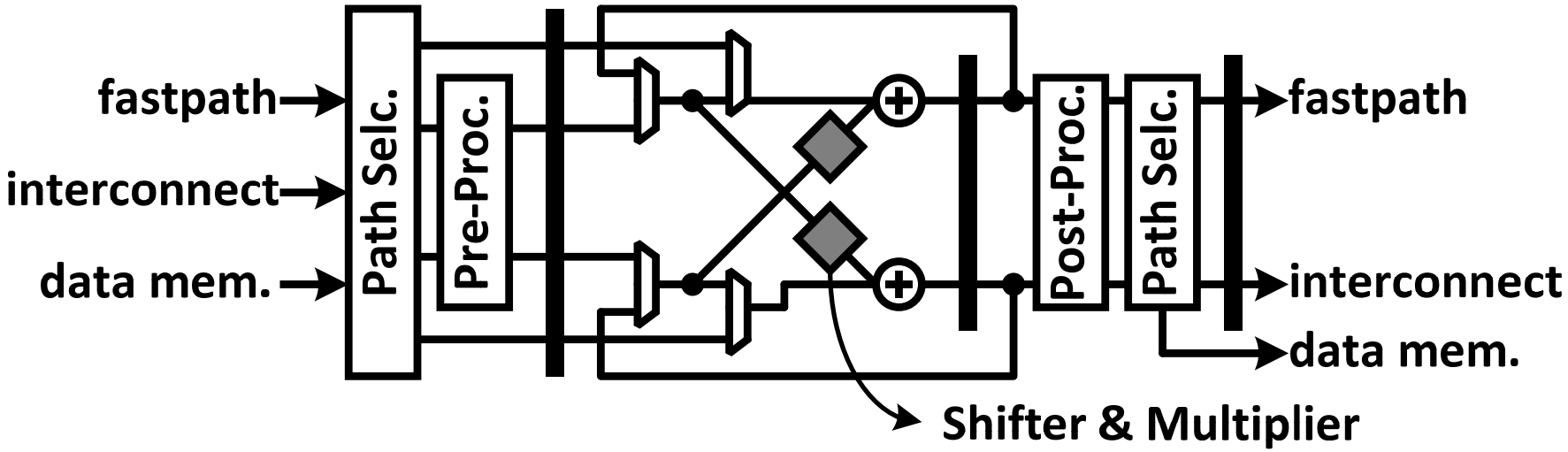
# 16-Core Universal DSP Kernel

- Built-in hierarchical network
- Local “fast-path” connections
- Also connects to the on-chip network

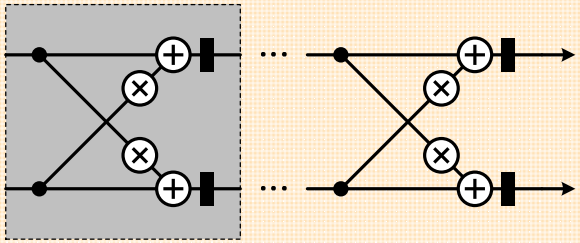


# 2 × 2 Butterfly Computing Element

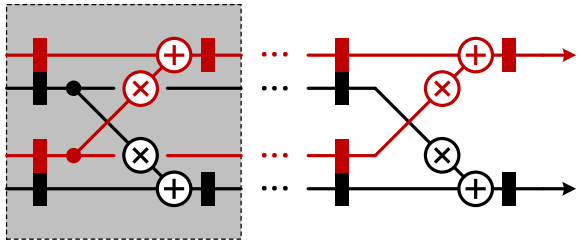
- Efficient for complex datapaths (Re + Im input)
- Implements algorithms by concatenating cores



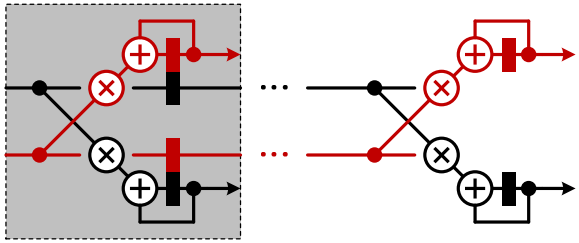
# Some Example DSP Functions



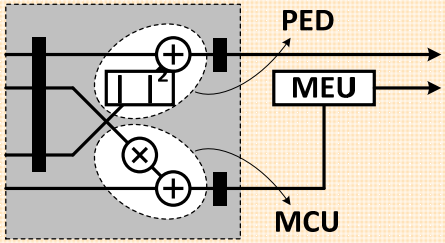
Lattice Filter



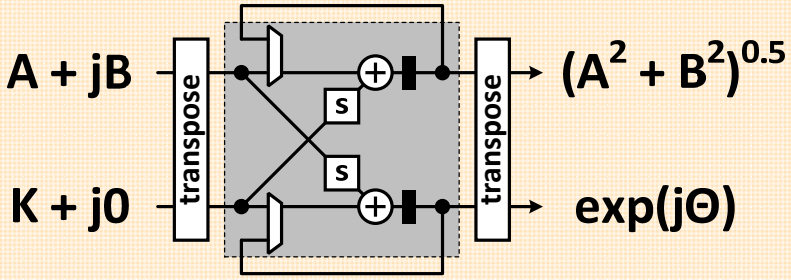
2-way FIR Filter



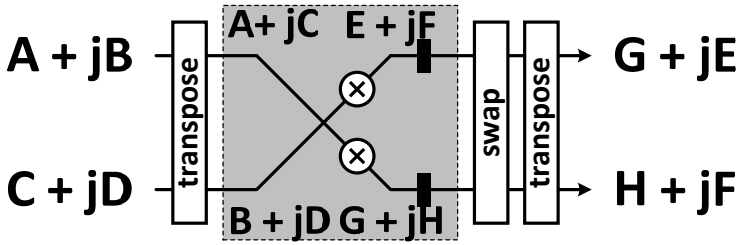
2-way ZF/MMSE Equalization




Sphere Decoder



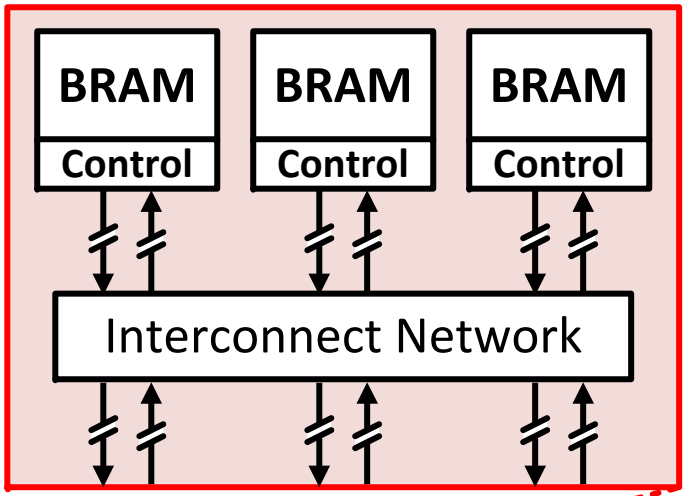

QR Factorization (scaling part)



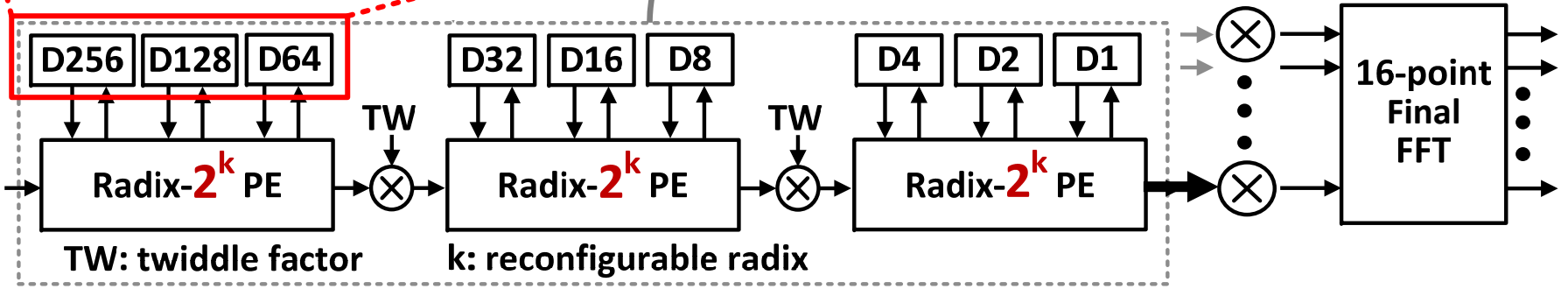

QR Factorization (rotation part)

# FFT Kernel

- **16 cores, each configurable for 4-512 point FFT**
  - Long delay lines realized using on-chip BRAM and CLBs



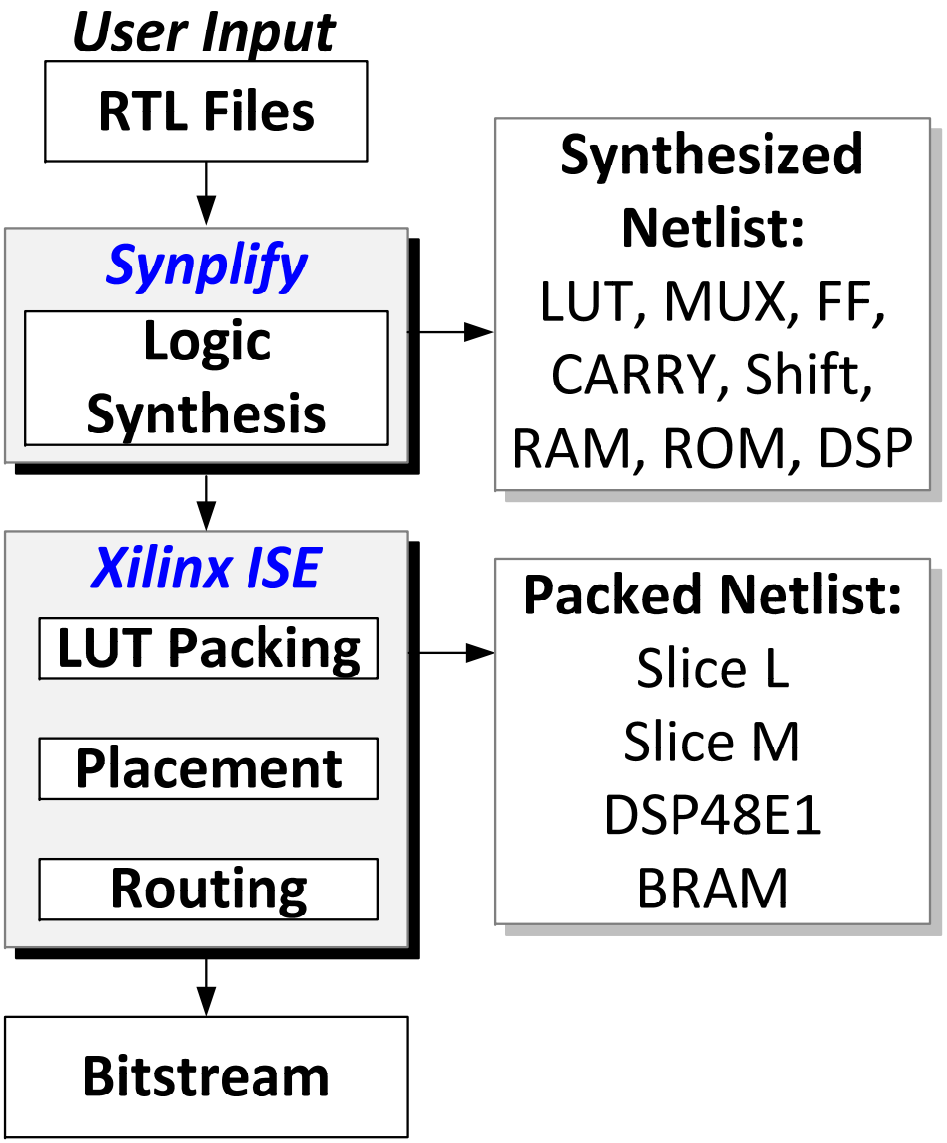

| FFT size | Pipeline FFT Radix (k) | | |
|---|---|---|---|
| 4 | - | - | 4 |
| 8 | - | - | 8 |
| 16 | - | 4 | 4 |
| 32 | - | 4 | 8 |
| 64 | - | 8 | 8 |
| 128 | 4 | 4 | 8 |
| 256 | 4 | 4 | 8 |
| 512 | 8 | 8 | 8 |



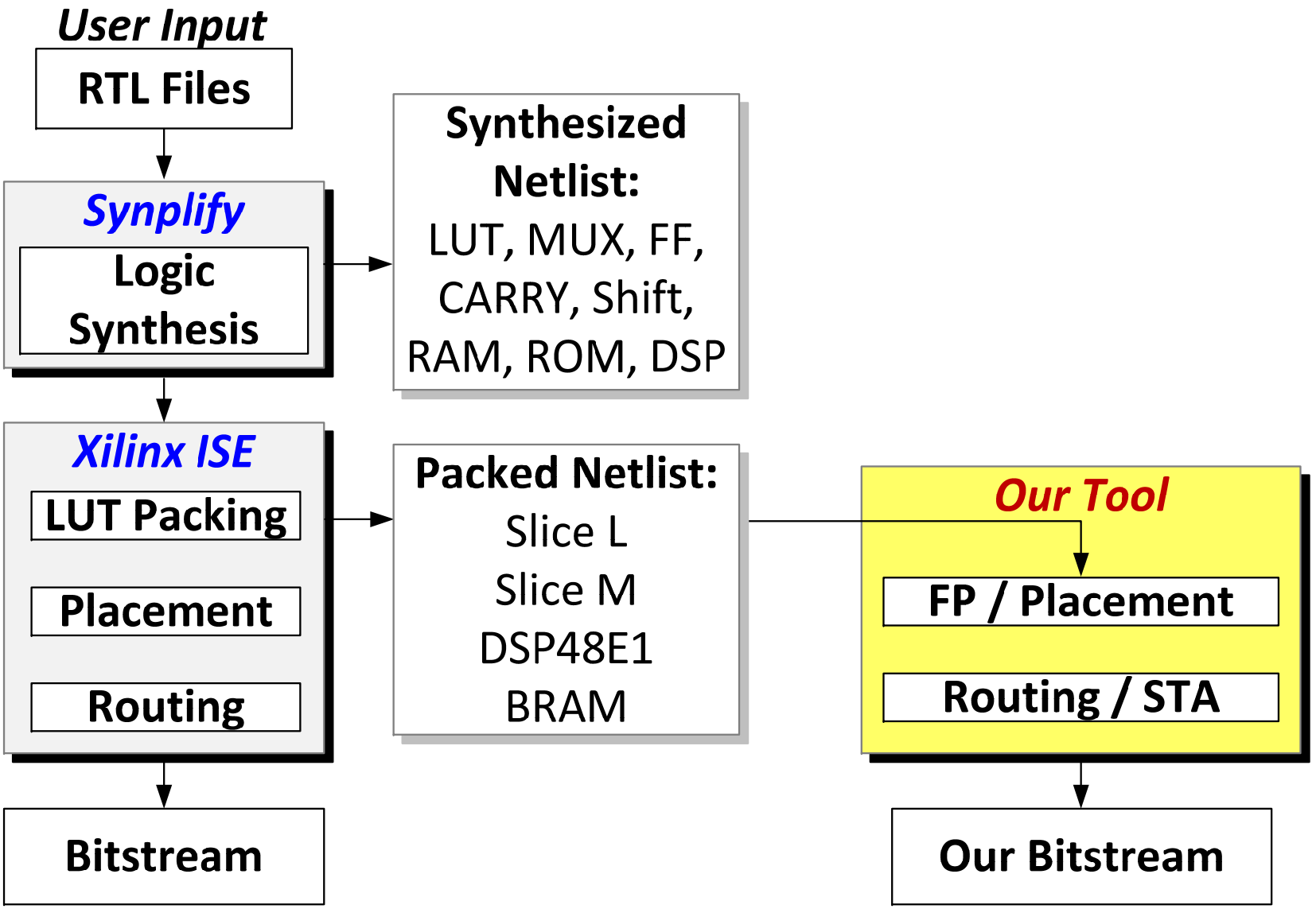
# Automated Mapping Flow

- **Leverage commercial tools**
  - Synthesis
  - LUT packing
- **Custom place & route tool**
  - Our interconnect
  - Coarse-grain kernels

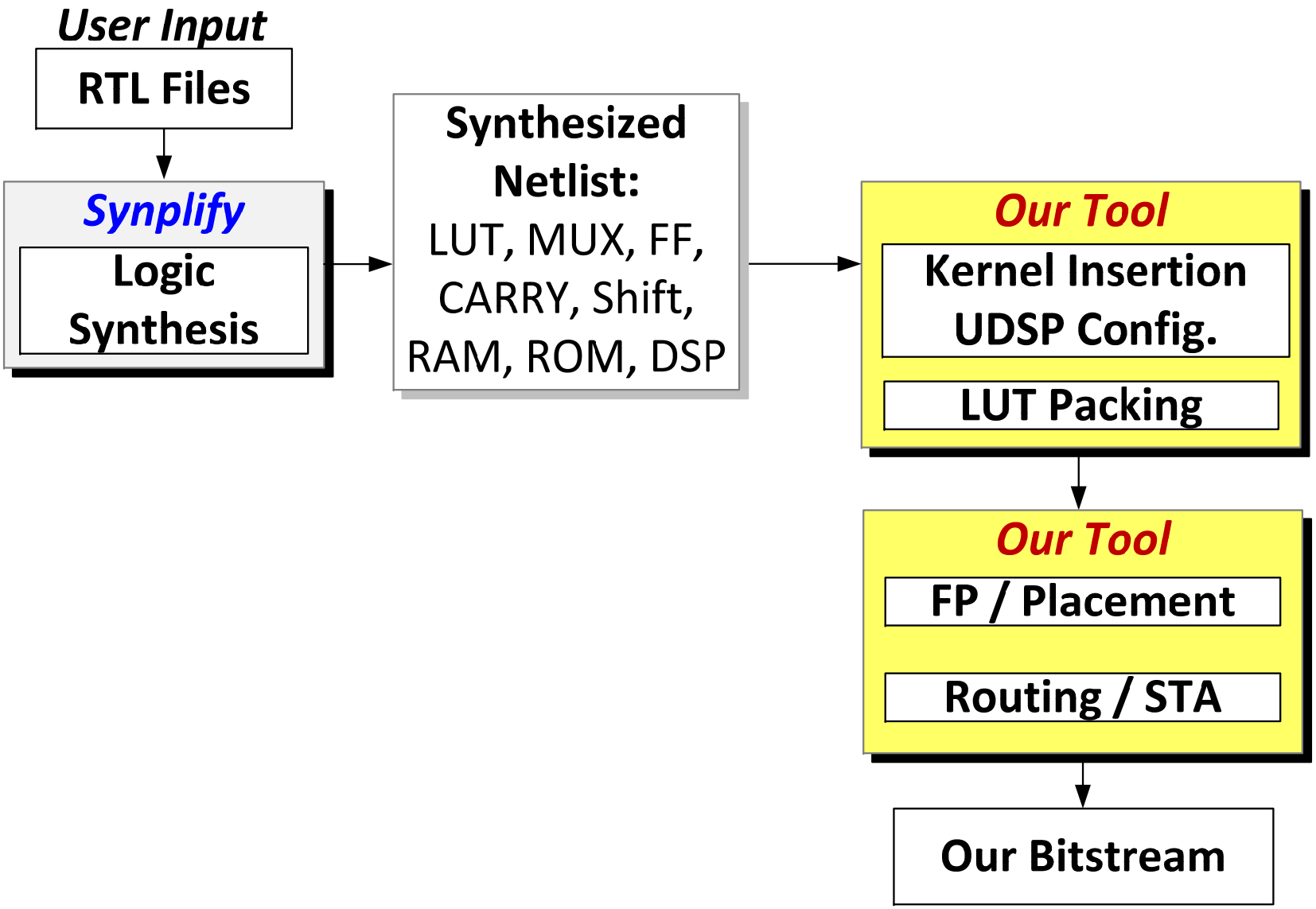
# Commercial FPGA Mapping



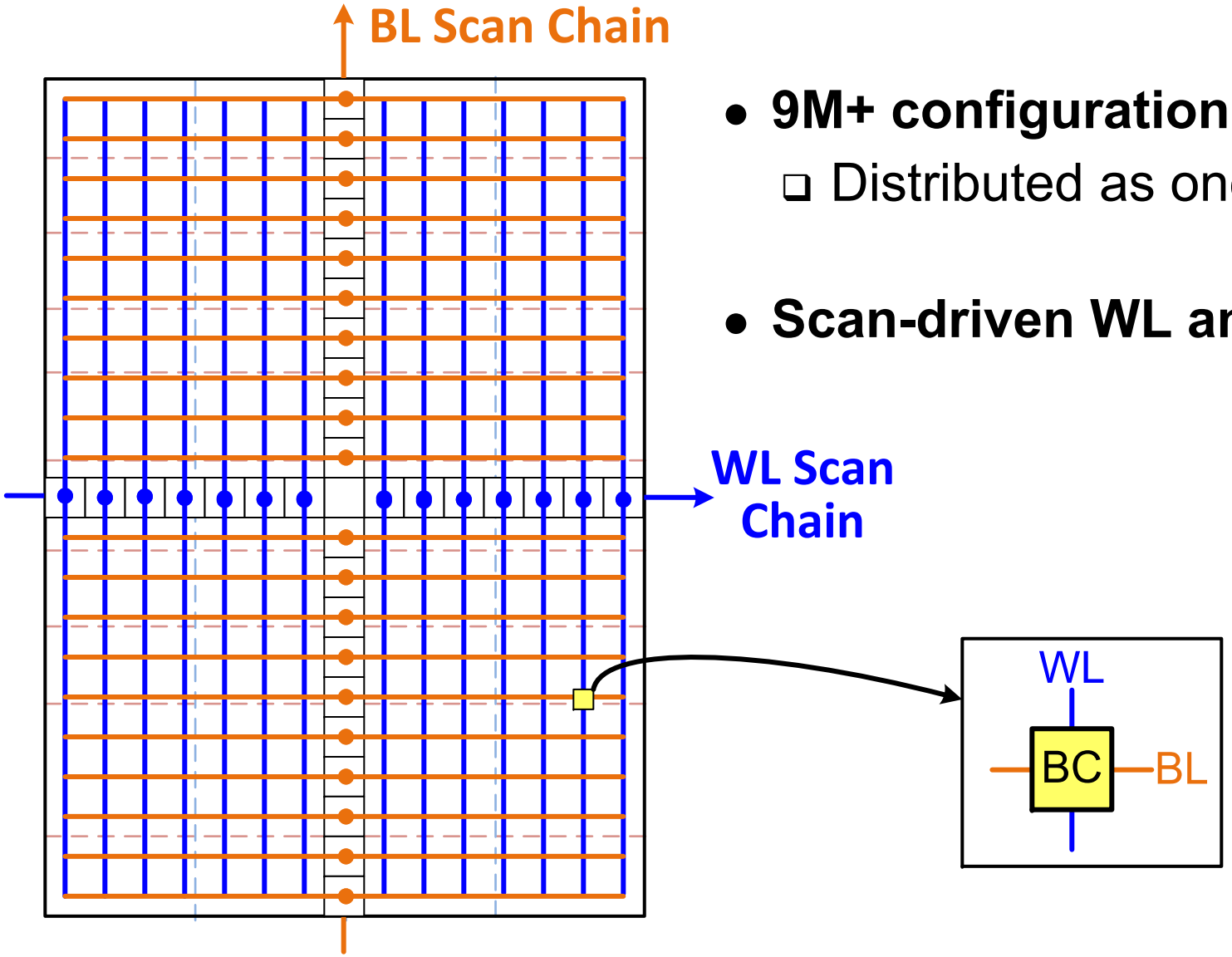
# Mode 1: Commercial Netlist P&R



# Mode 2: Kernel Insertion P&R

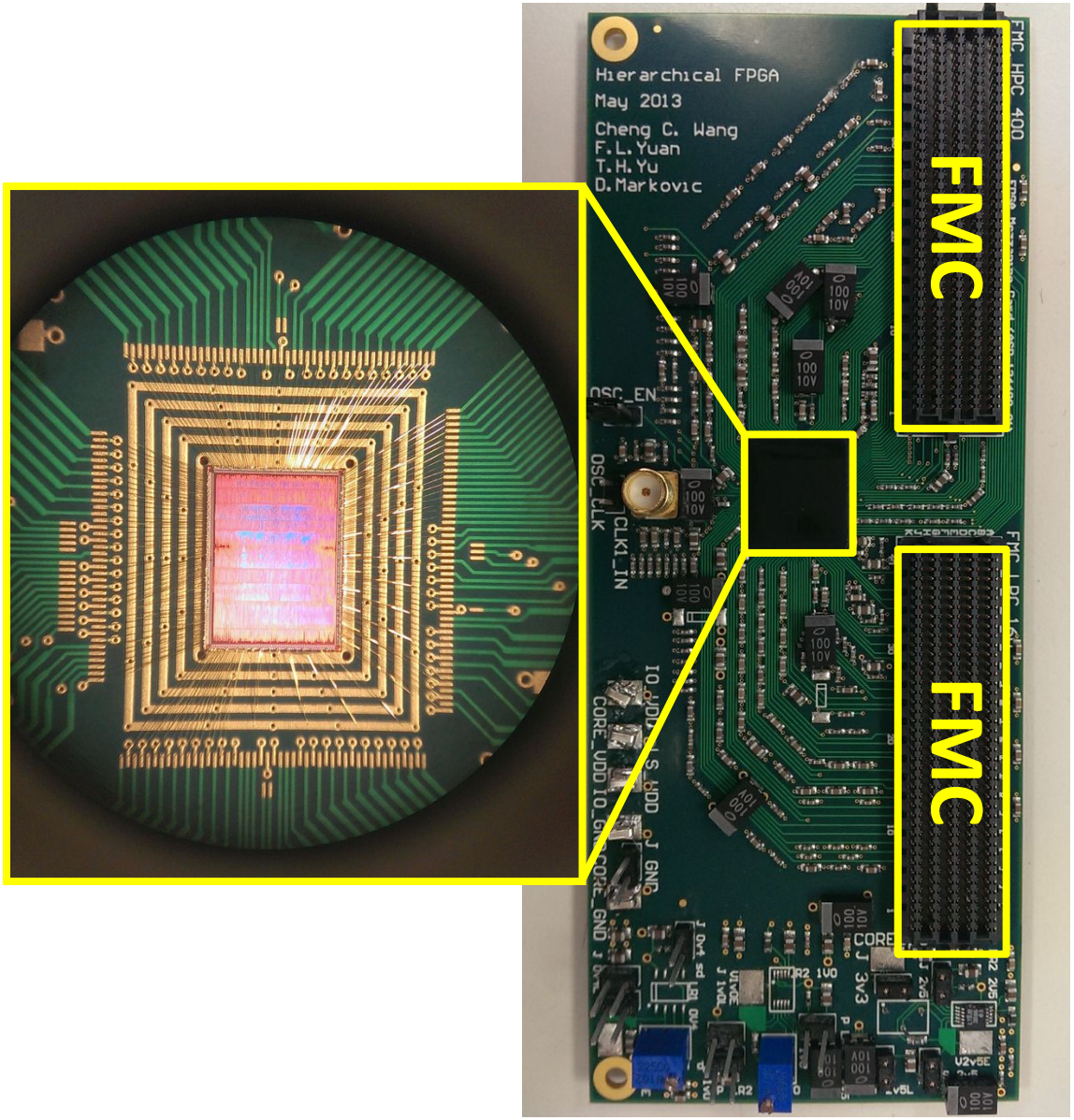


# Configuration Circuitry




- **9M+ configuration bits**
  - Distributed as one array
- **Scan-driven WL and BL**

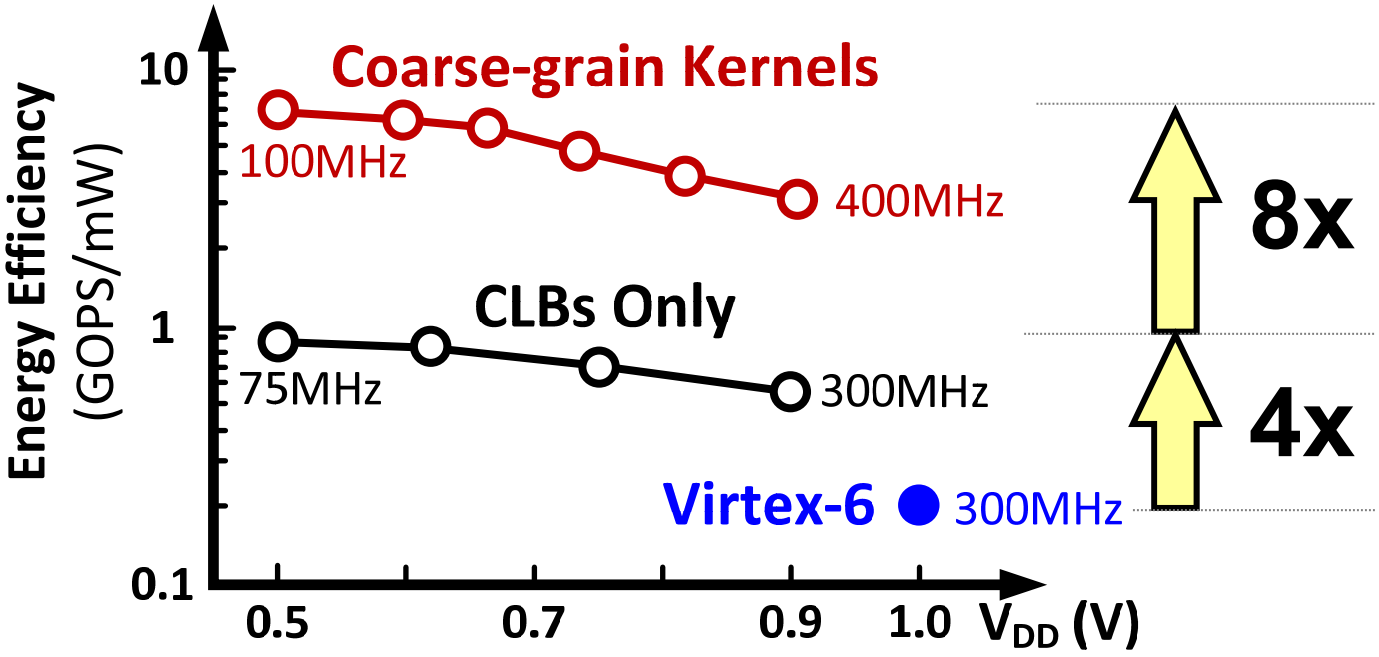
# Chip-on-Board for Chip Testing




- **171 user I/Os bonded**
- **2 FMC Connectors**
  - Interface to Kintex-7 test FPGA

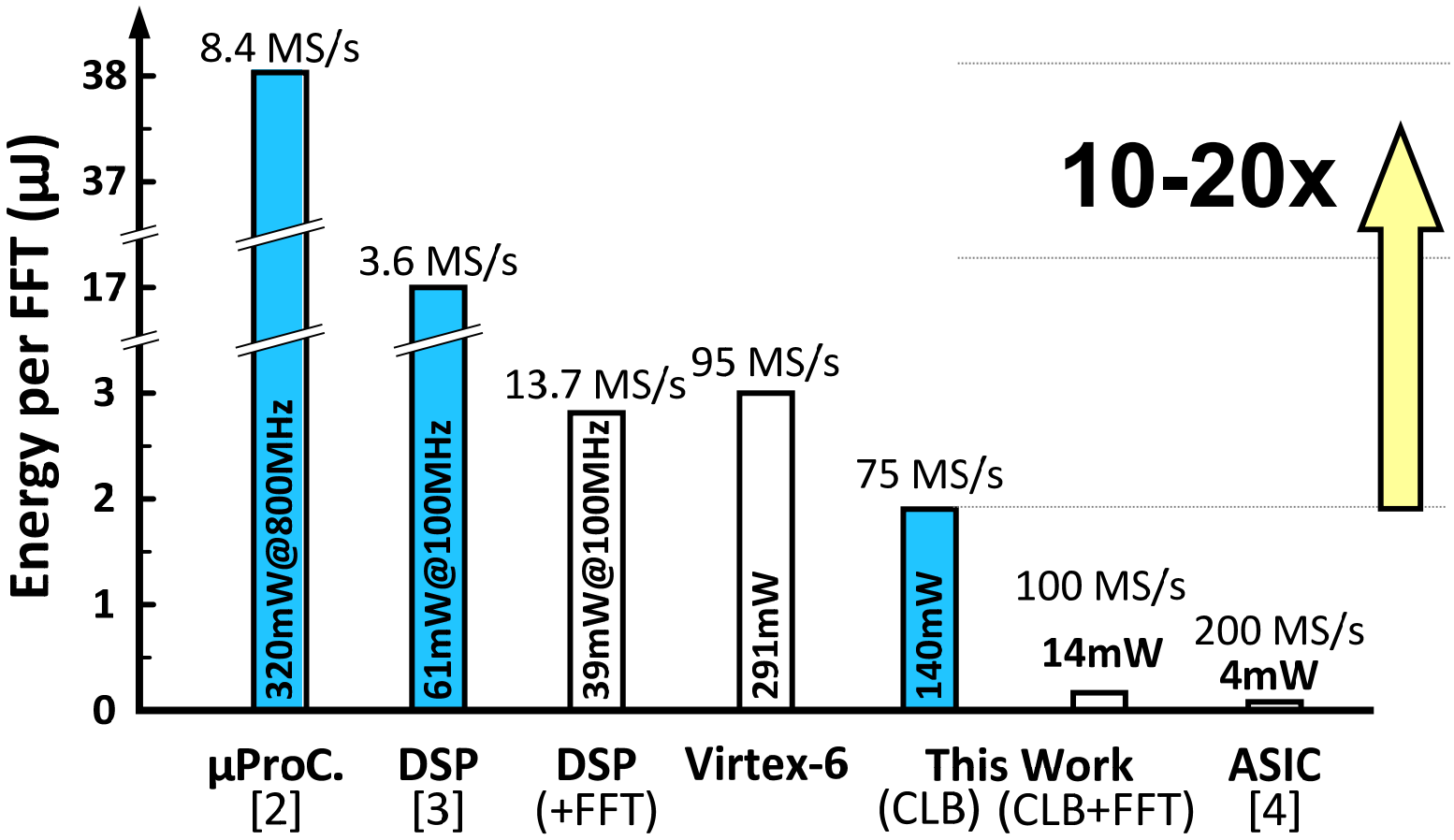
# Measurement Results: FIR Filter

- **4x more efficient than existing FPGAs**
  - Up to 0.86 GOPs/mW at 0.5V
- **Additional 8x gain using UDSP kernels**



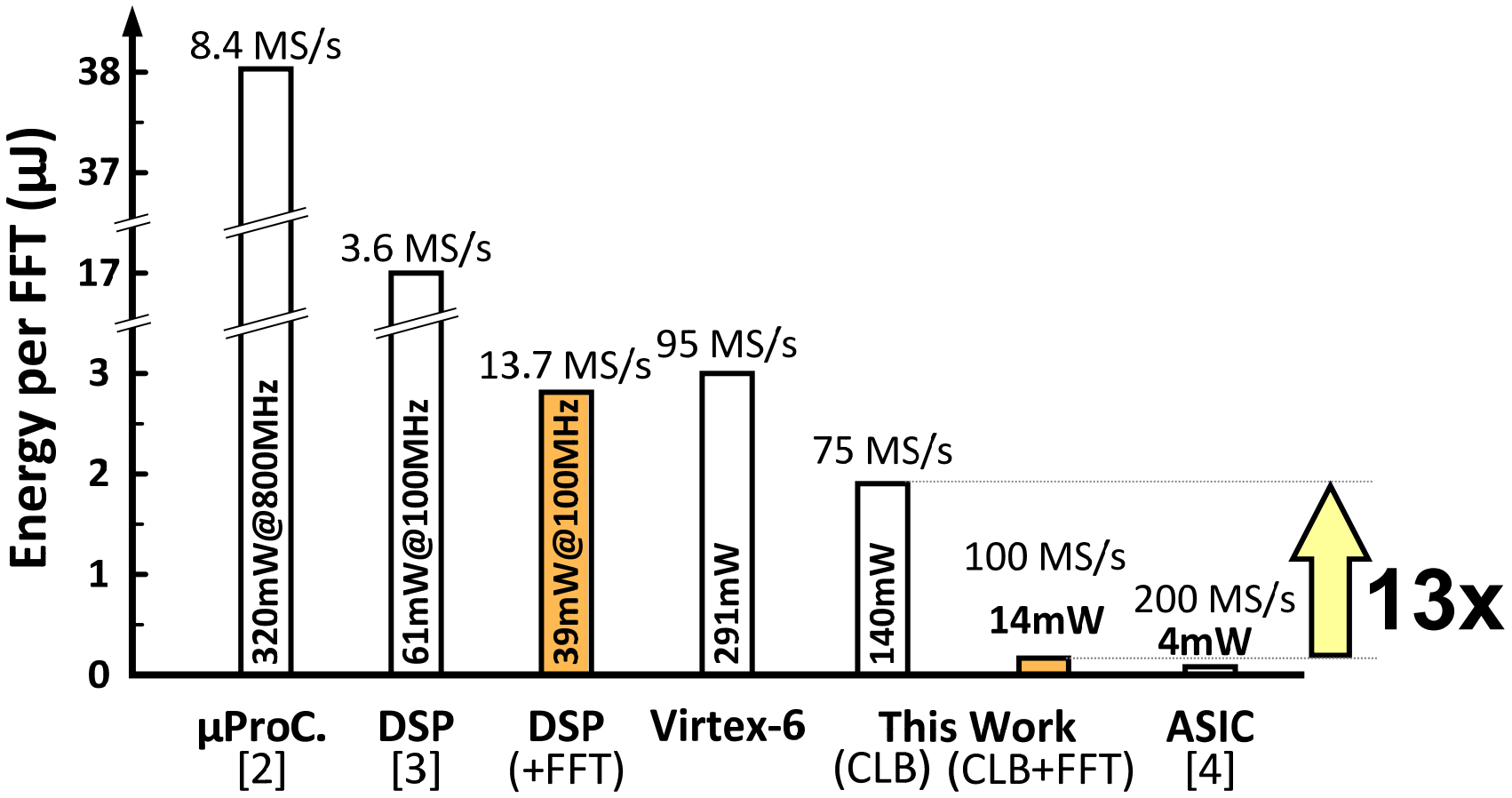
# Measurement Results: 1024-pt FFT

- 10-20x more efficient than DSPs and processors



# Measurement Results: 1024-pt FFT

- 10-20x more efficient than DSPs and processors
- Additional 13x gain using FFT kernel



# This Work (CLBs Only) vs. ASIC

| Design \ Resources | Utilization | | | Energy Eff. gap vs. ASIC (norm.) |
|---|---|---|---|---|
| | L/M | DSP | BRAM | |
| | 2760 | 42 | 16 | |
| 256-point MIMO FFT [5] | 916 | 4 | 4 | **12×** |
| K-best + depth first Sphere Decoder | 1255 | 0 | 0 | **15×** |
| Software-Defined Radio Tx Front End | 1594 | 38 | 0 | **13×** |
| 802.11a Rx Baseband [6] | 1824 | 0 | 6 | **15×** |

**~30-60x area penalty vs. ASIC**

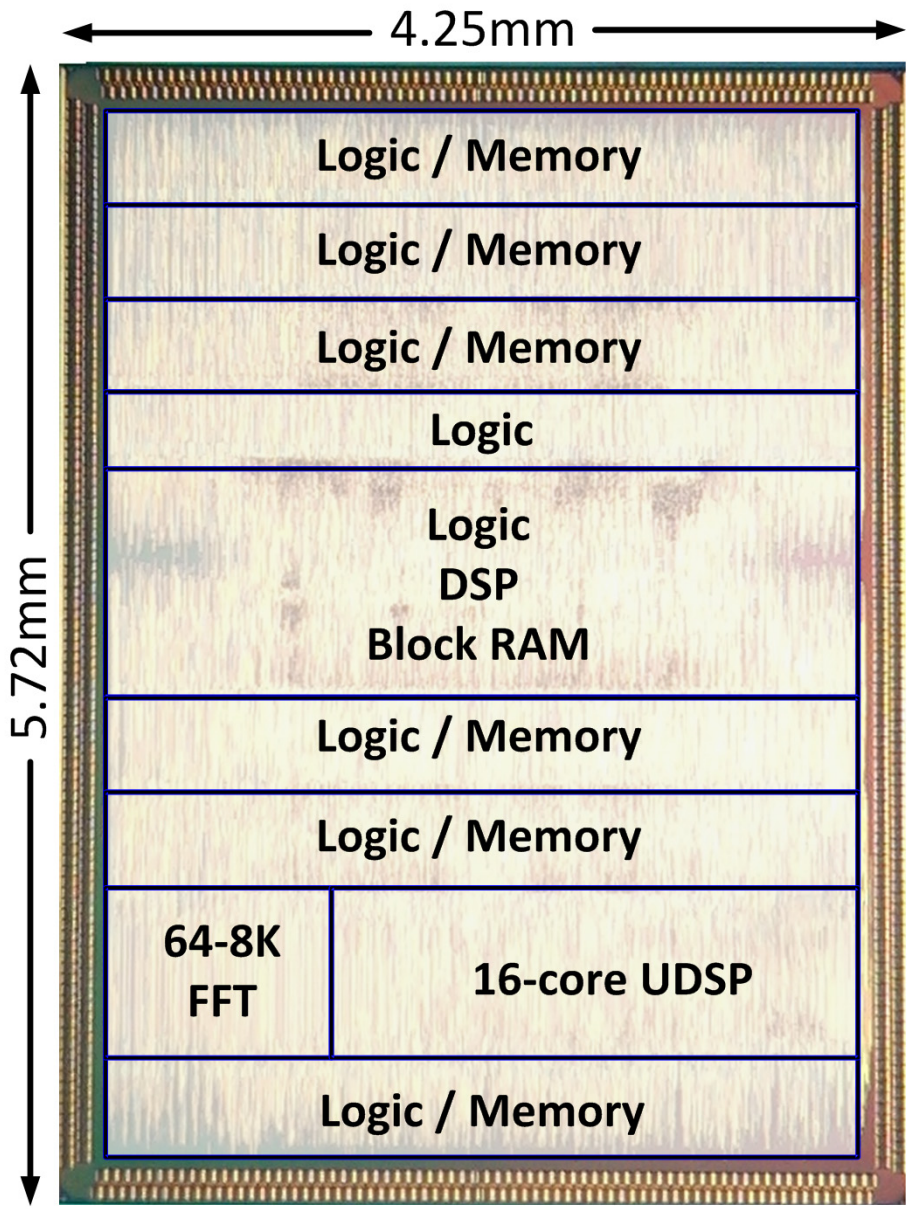
# This Work (CLBs + Kernels) vs. ASIC

- **Offload computationally-intensive blocks to kernels**
  - Remaining logic uses fine- and medium- grain CLBs

| Resources / Design | Utilization | | | | | Energy Eff. gap vs. ASIC (norm.) |
|---|---|---|---|---|---|---|
| | L/M | DSP | BRAM | FFT | UDSP | |
| | 2760 | 42 | 16 | 1 | 16 | |
| Software-Defined Radio Tx Front End | 664 | 26 | 0 | 0 | 12 | **4×** |
| 802.11a Rx Baseband [6] | 701 | 0 | 6 | 1 | 5 | **5×** |

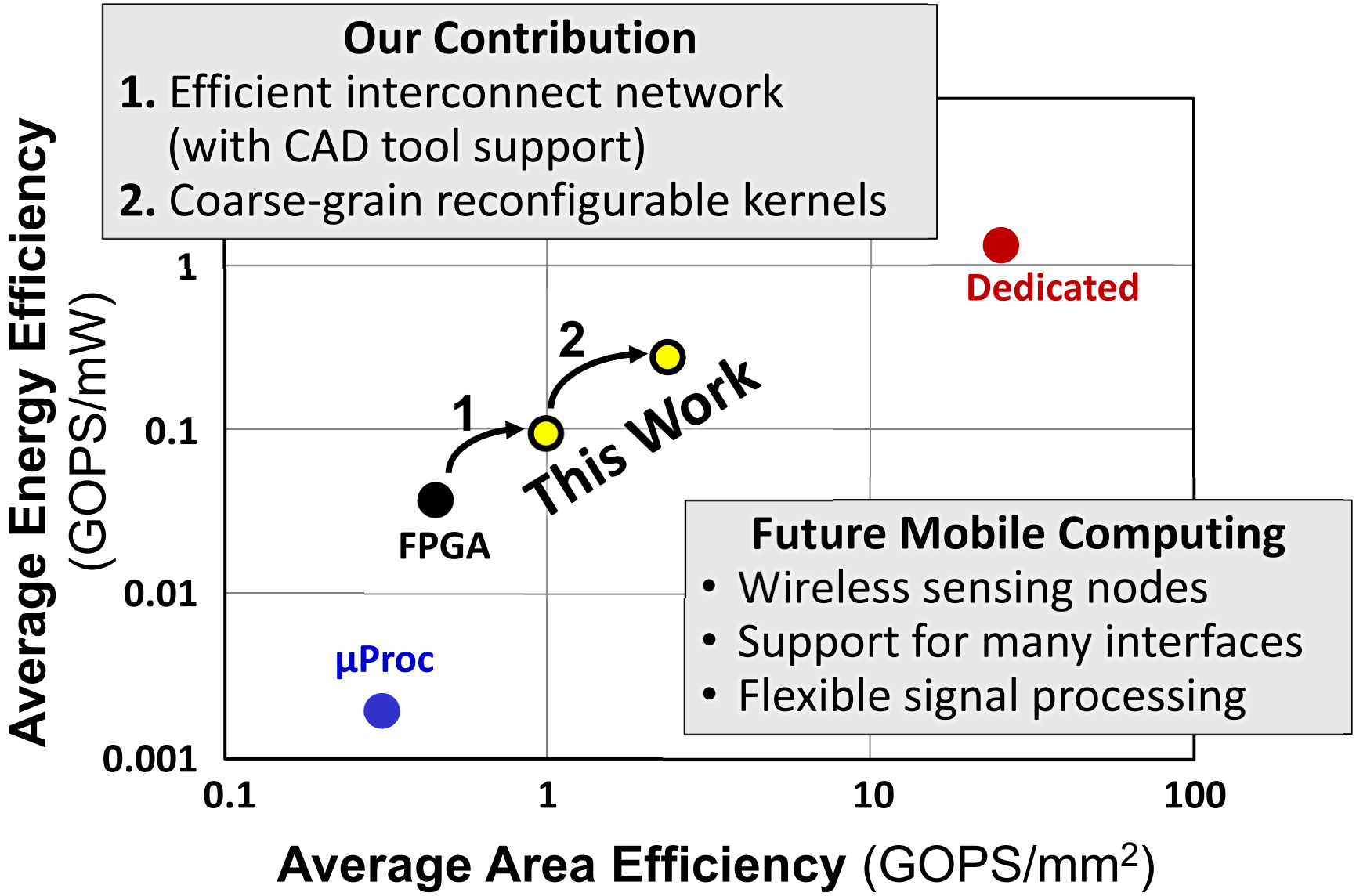
**~12-30x area penalty vs. ASIC**

# Chip Micrograph and Summary




| | |
|---|---|
| Technology | 40nm 1P10M CMOS |
| Core $V_{DD}$ | 0.5 to 0.9V |
| Frequency | ≤ 400 MHz |
| Energy Efficiency | ≤ 0.86 GOPS/mW (CLBs)<br>≤ 7 GOPS/mW (Kernels) |
| I/Os | 350 bidirectional<br>(171 bonded) |
| Core Size | 3.8mm × 5.3mm |
| Transistor Count | 143 million |
| L/M CLB | 2176 Logic<br>576 Memory |
| DSP CLB | 42 MAC/SIMD Processor |
| Block RAMs | 16 36Kb BRAM |
| Config. Bits | 9,279,168 |

# Conclusion: Efficiency + Flexibility



# An 821MHz 7.9Gb/s 7.3pJ/b/iteration Charge-Recovery LDPC Decoder

Tai-Chuan Ou, Zhengya Zhang,
Marios Papaefthymiou

Department of Electrical Engineering
and Computer Science

University of Michigan, Ann Arbor

# Motivation

- Digital communication applications require both low power and high performance
- Low-density parity-check (LDPC) decoding is computation-intensive and power-hungry
- Charge-recovery logic has the potential to achieve both requirements
- To demonstrate this potential, we implement a charge-recovery LDPC decoder

# Design Highlights

- Charge-recovery LDPC decoder in 65nm CMOS for a (576,480) LDPC code (IEEE 802.16e)
- With 16 on-chip inductors, the test chip resonates at 821MHz, functional up to 1.05GHz
- 1.7× improvement in energy efficiency over previous commercial-strength LDPCs
- 32× more transistors than previous charge-recovery designs, achieved by proposed semi-custom design methodology

# Previous Charge-Recovery Designs

| | ISSCC '06 | ESSCIRC '09 | VLSI '09 | ASSCC '11 | This Work | |
|---|---|---|---|---|---|---|
| Technology | 0.13µm | 0.13µm | 0.13µm | 65nm | 65nm | |
| Application | Chains of test gates | FIR | FIR | Processing node for LDPC | LDPC | |
| Frequency Range (MHz) | 700 – 1,100 | 365 - 600 | 5 - 187 | 404 - 609 | 621 – 1,049 | |
| Active Area ($mm^2$) | 0.02 | 0.34 | 0.38 | 0.04 | 1.54 | |
| Transistor Count (x1000) | N/A | N/A | 41 | 1.64 | 1,297 | 32× |
| Gate Count | 1,680 | 3,330 | N/A | N/A | 55,305 | 17× |

- Previous work is limited to small designs

# Previous LDPC Designs

| | JSSC '11 | ASSCC '11 | JSSC '12 | ISSCC '13 | This Work |
|---|---|---|---|---|---|
| Technology | 0.13µm | 65nm | 65nm | 65nm | 65nm |
| Code Length | 576~2,304 | 576~2,304 | 672 | 32 | 576 |
| Frequency (MHz) | 214 | 110 | 197 | 240 | 821 |
| Throughput (Gbps) | 0.847 | 1.056 | 5.79 | 0.38 | 7.882 |
| Energy Efficiency (pJ/bit/iteration) | 13.54 | 21.8 | 12.48 | 2.08 | 7.32 |

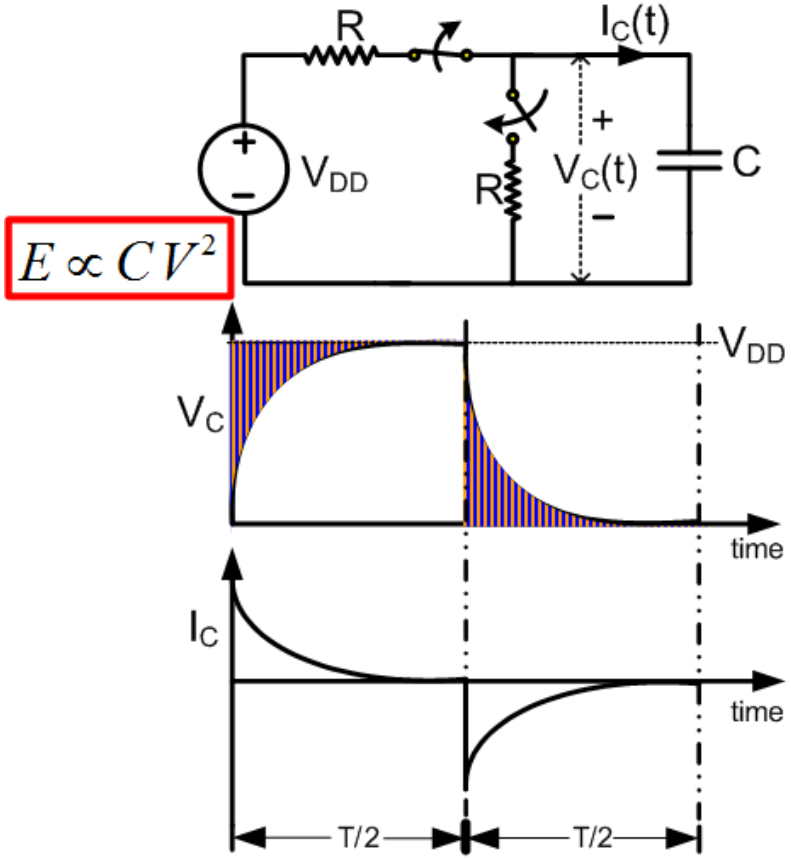
1.7×

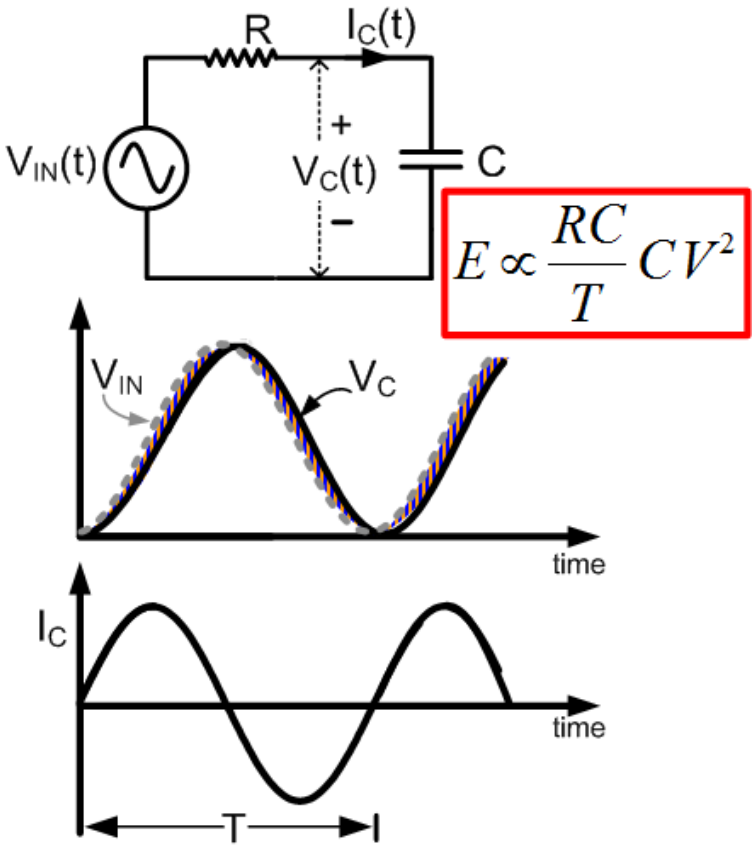
- Our goal: Achieve high energy efficiency while maintaining high throughput

# Outline

- Motivation and Previous Work
- **Charge-Recovery Logic**
- LDPC Decoder Design and Architecture
- Power-Clock Distribution Network
- Design Methodology
- Test Chip Results
- Conclusion

# Charge Recovery Logic Basics

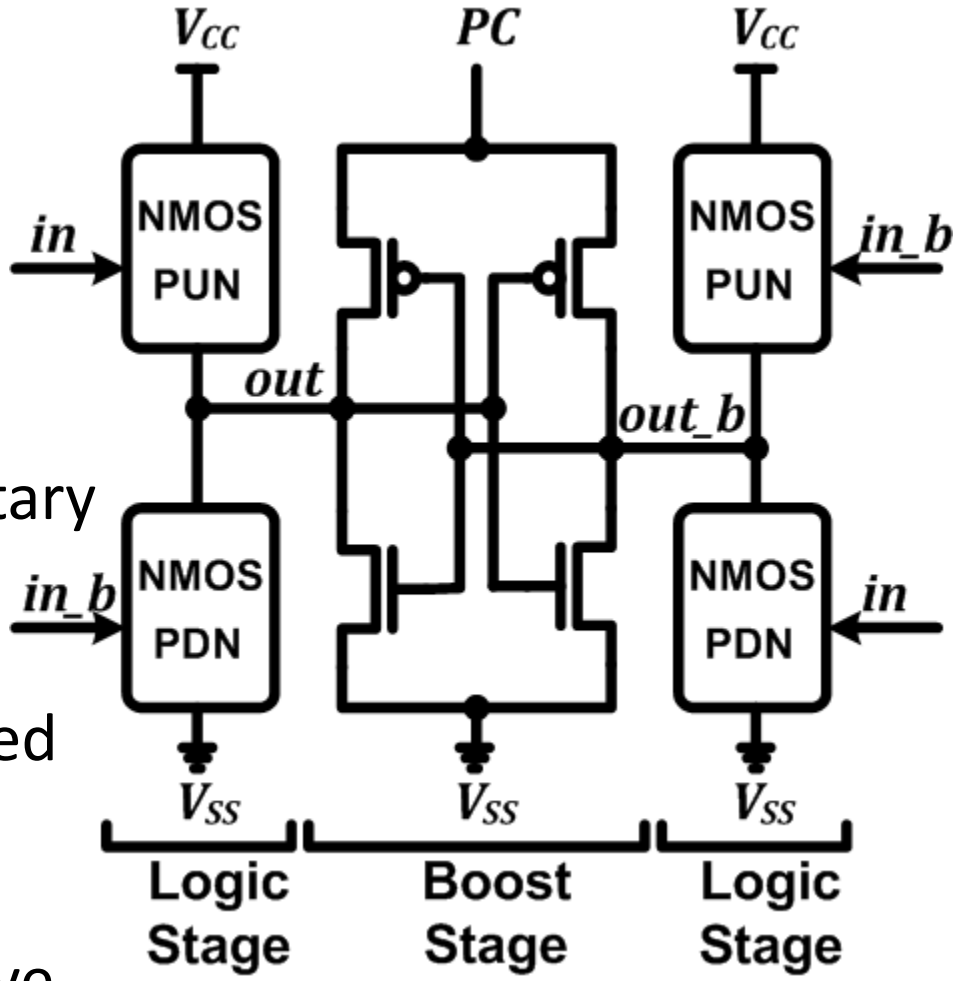


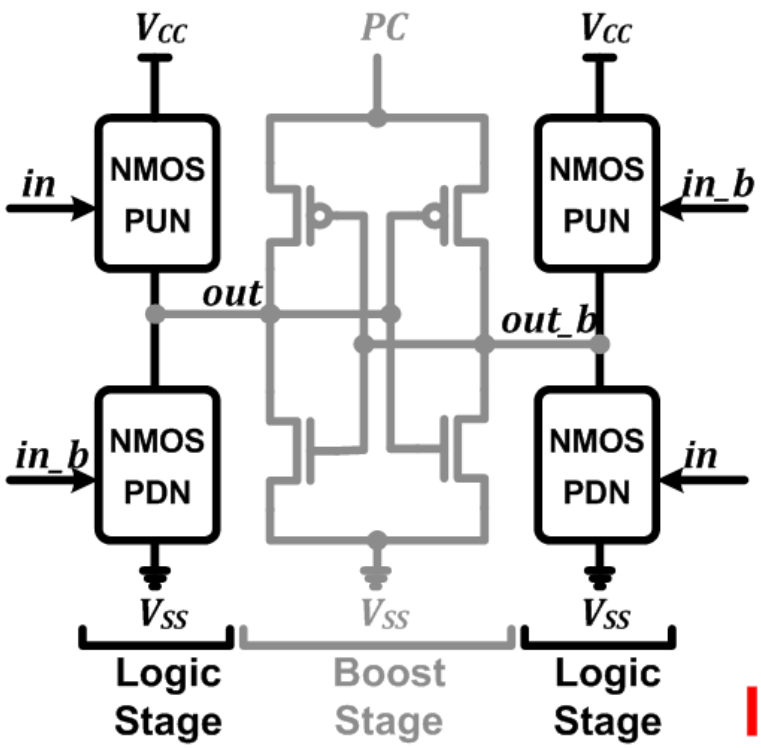

- Conventional CMOS
  - Supply provides charge at full voltage $V_{DD}$
  - No energy exchanged between supply and load
- Charge recovery logic
  - Supply transitions gradually, low voltage drop across R
  - Energy exchanged between supply and load

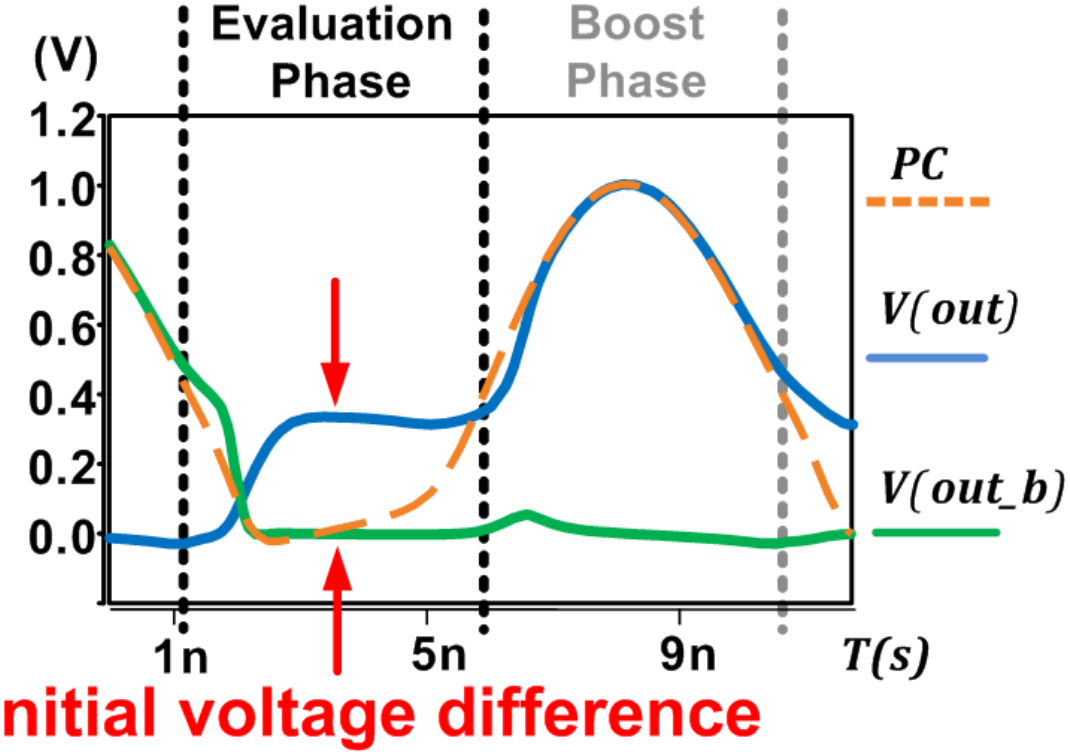
# Boost Logic (BL)

- Boost logic combines efficient signal boosting with charge recovery [ISSCC '06, ESSCIRC '09, VLSI '09]
  - Logic Stage: complementary pull-up and pull-down network (PUN / PDN)
  - Boost Stage: cross-coupled inverters supplied by power-clock (PC)
  - PC: generated by inductive elements



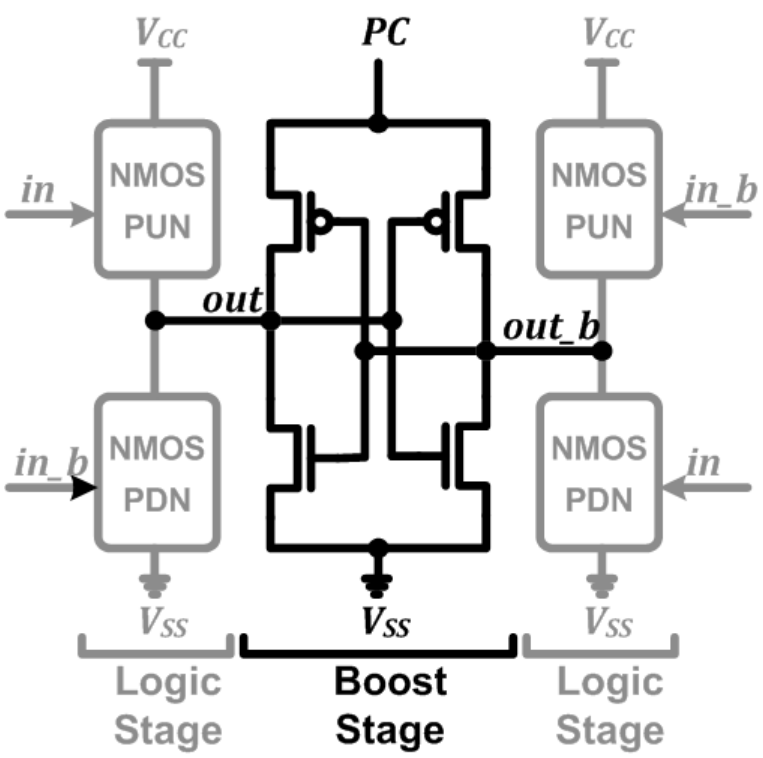
# Two-Phase Operation: Evaluation

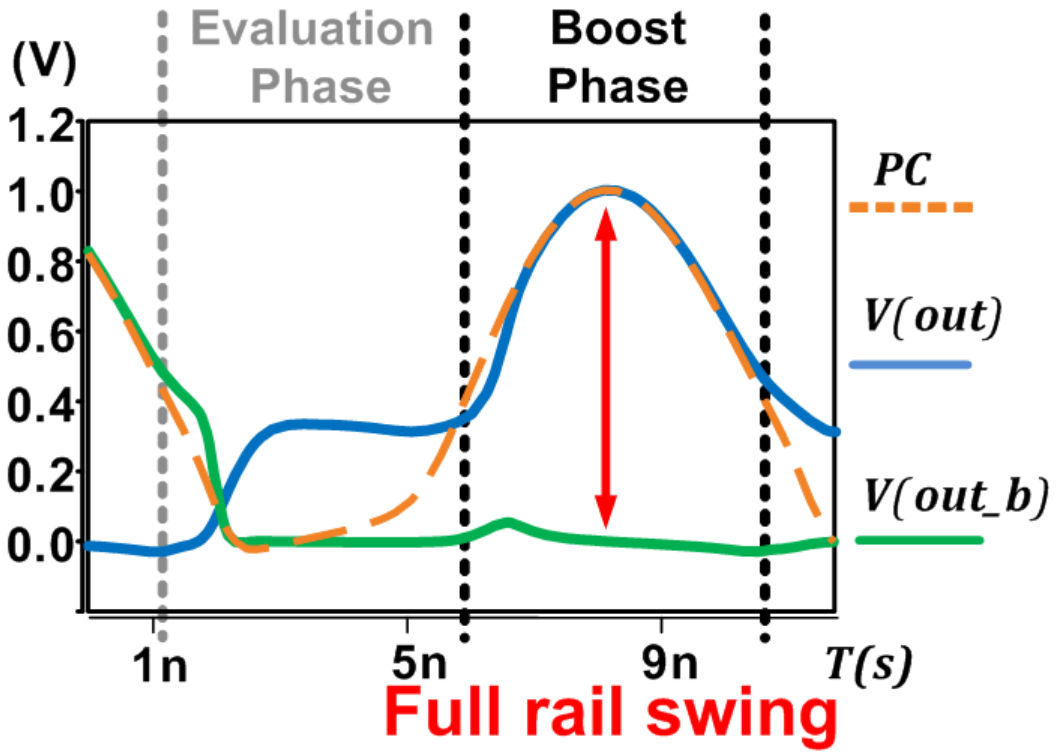



- Boost Stage is off, and Logic Stage is active
- Perform functional evaluation and develop an initial voltage difference at the output nodes ($out$ and $out\_b$)

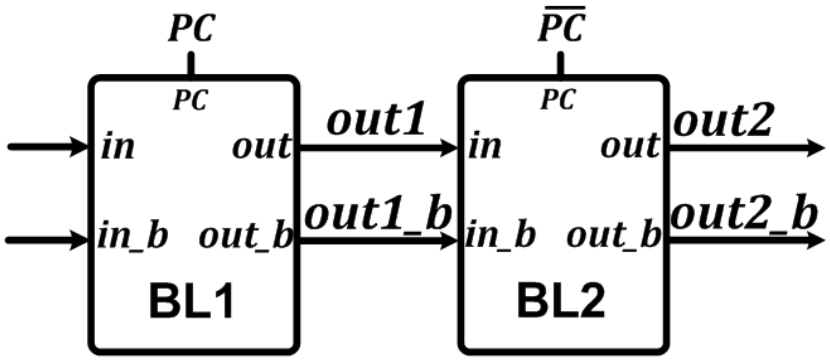
# Two-Phase Operation: Boost

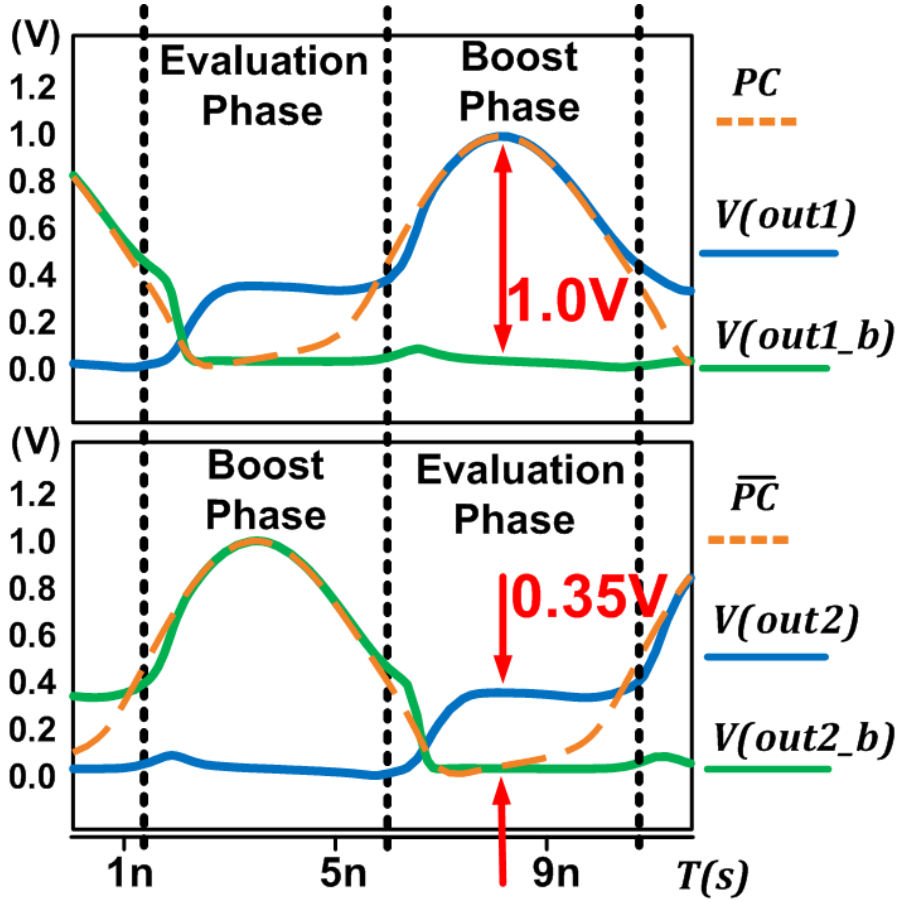



- As PC rises, Boost Stage turns on. Output nodes track PC and are amplified to full-rail signals
- As PC falls, charge at the output nodes is recovered through PC

# Cascading BL Gates

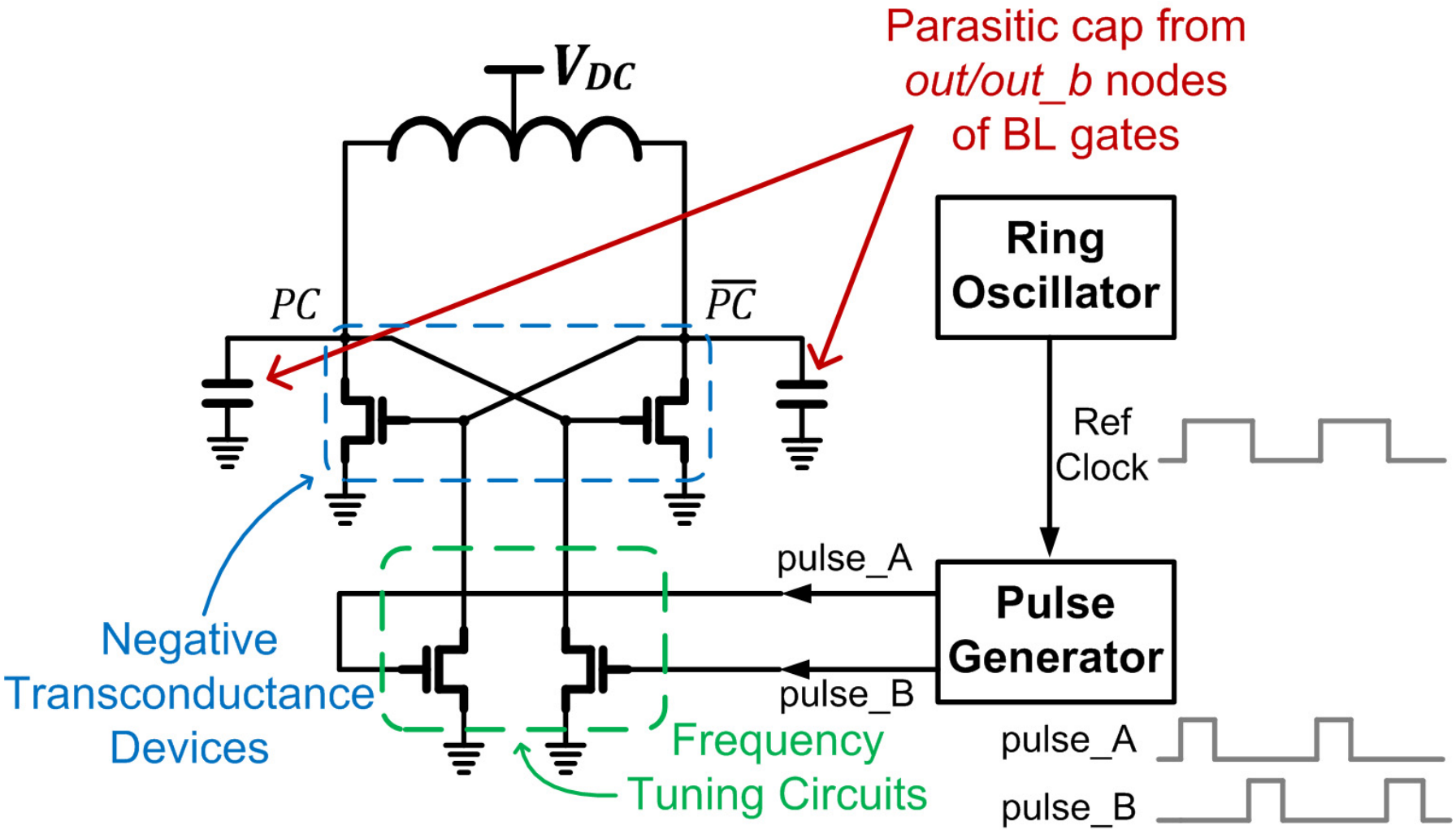



- Use alternate clock phases to cascade BL gates
- Full rail inputs from previous phase provide gate overdrive to the current phase

# On-Chip Power-Clock Generator

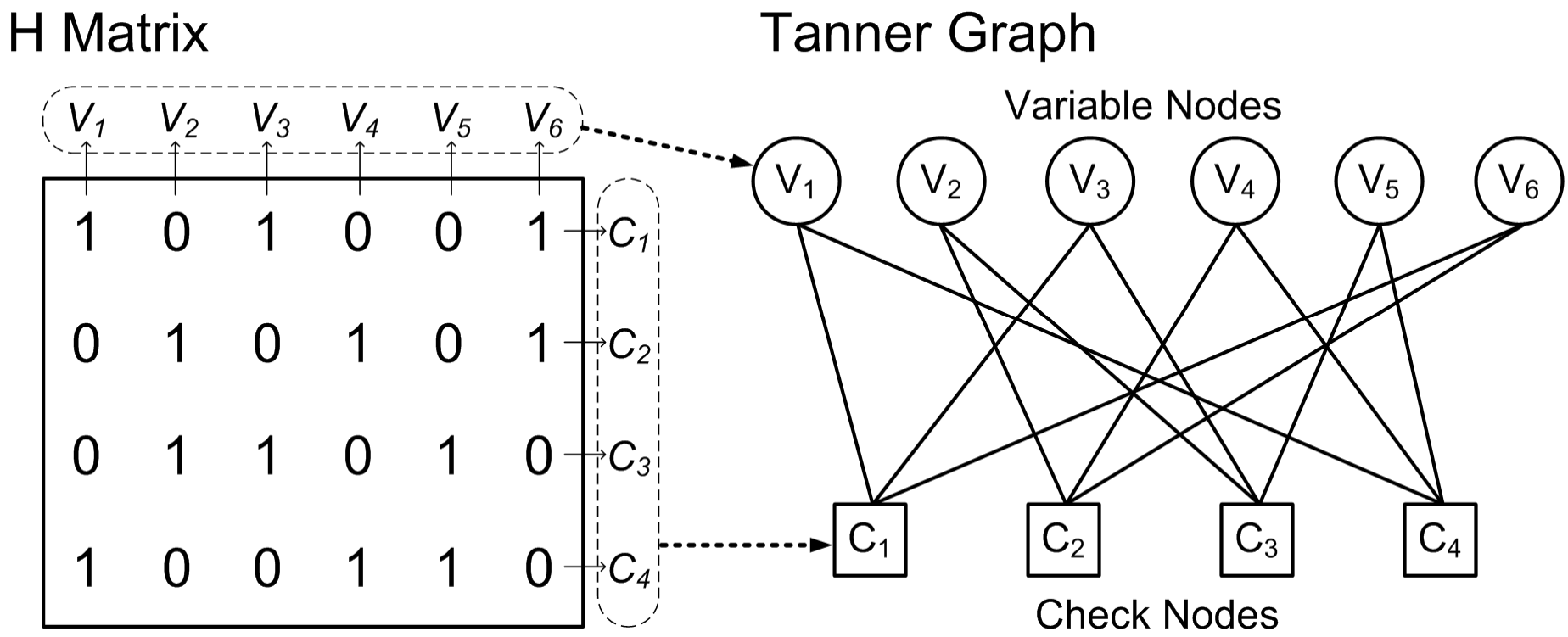
- Adaptation of blip generator [ISCAS '96]



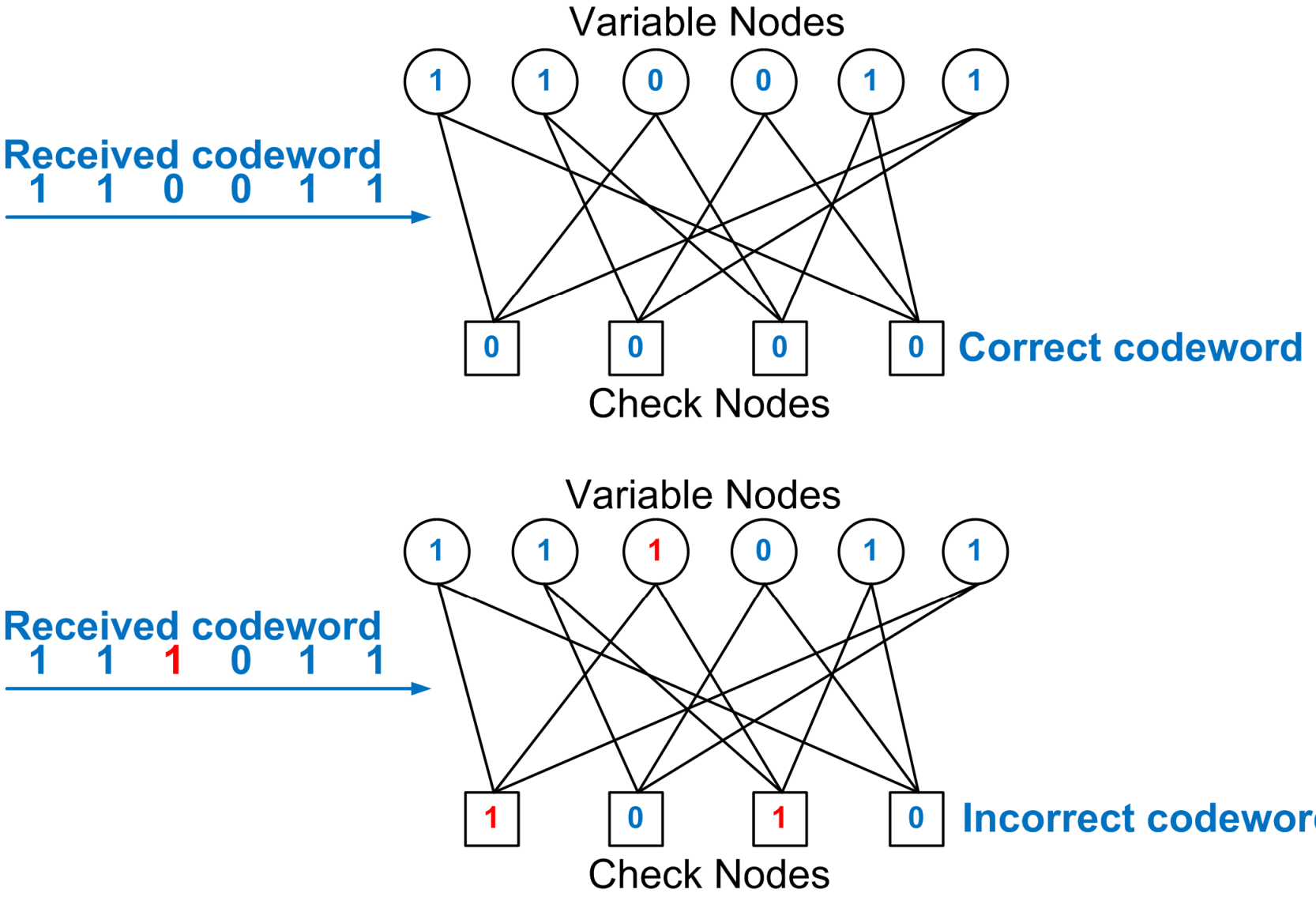
# Outline

- Motivation and Previous Work
- Charge-Recovery Logic
- **LDPC Decoder Design and Architecture**
- Power-Clock Distribution Network
- Design Methodology
- Test Chip Results
- Conclusion

# LDPC Introduction



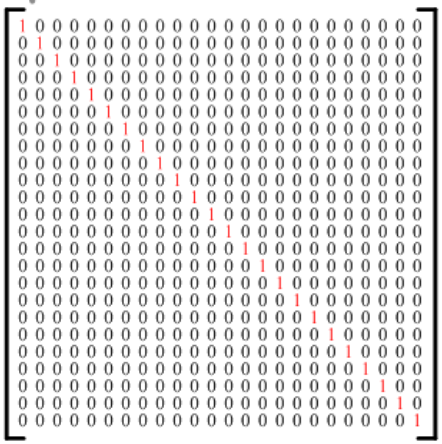
# Message Passing Decoding

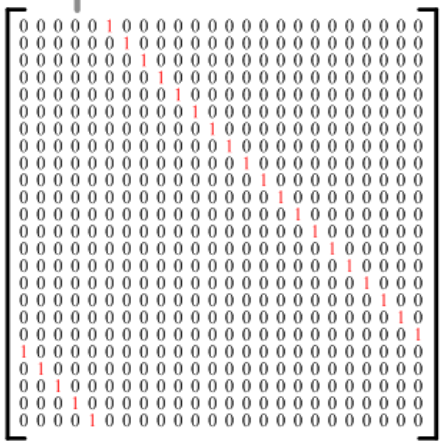


# LDPC H-Matrix

- IEEE 802.16e 576-Bit, Rate-5/6 H-Matrix

| | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 6 | 13 | - | 11 | 1 | - | 22 | 21 | 2 | 21 | 13 | 20 | 8 | 1 | 0 | 9 | 5 | 1 | 19 | 20 | 0 | - | - |
| - | 1 | - | 9 | 10 | 11 | 3 | 19 | 11 | - | 10 | 5 | 3 | 17 | 3 | 18 | 0 | 11 | 12 | 0 | 0 | 0 | 0 | - |
| 12 | 20 | 20 | 1 | 16 | - | 5 | - | 7 | 6 | 22 | 15 | 20 | 2 | 21 | 19 | 15 | 22 | 16 | 3 | - | - | 0 | 0 |
| 12 | - | 12 | 3 | - | 9 | 3 | 2 | 2 | 5 | 13 | 22 | 7 | 23 | 14 | 7 | 21 | 23 | 2 | 16 | 20 | - | - | 0 |

0: 

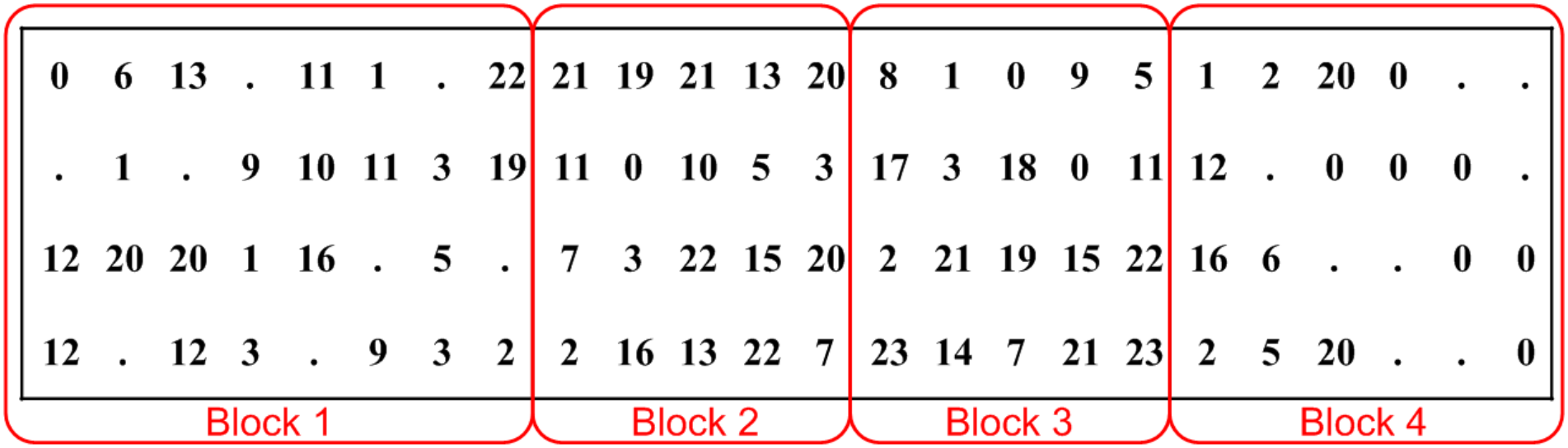
5: 

24×24 zero matrix

Each number represents a 24×24 circular right-shifted identity matrix

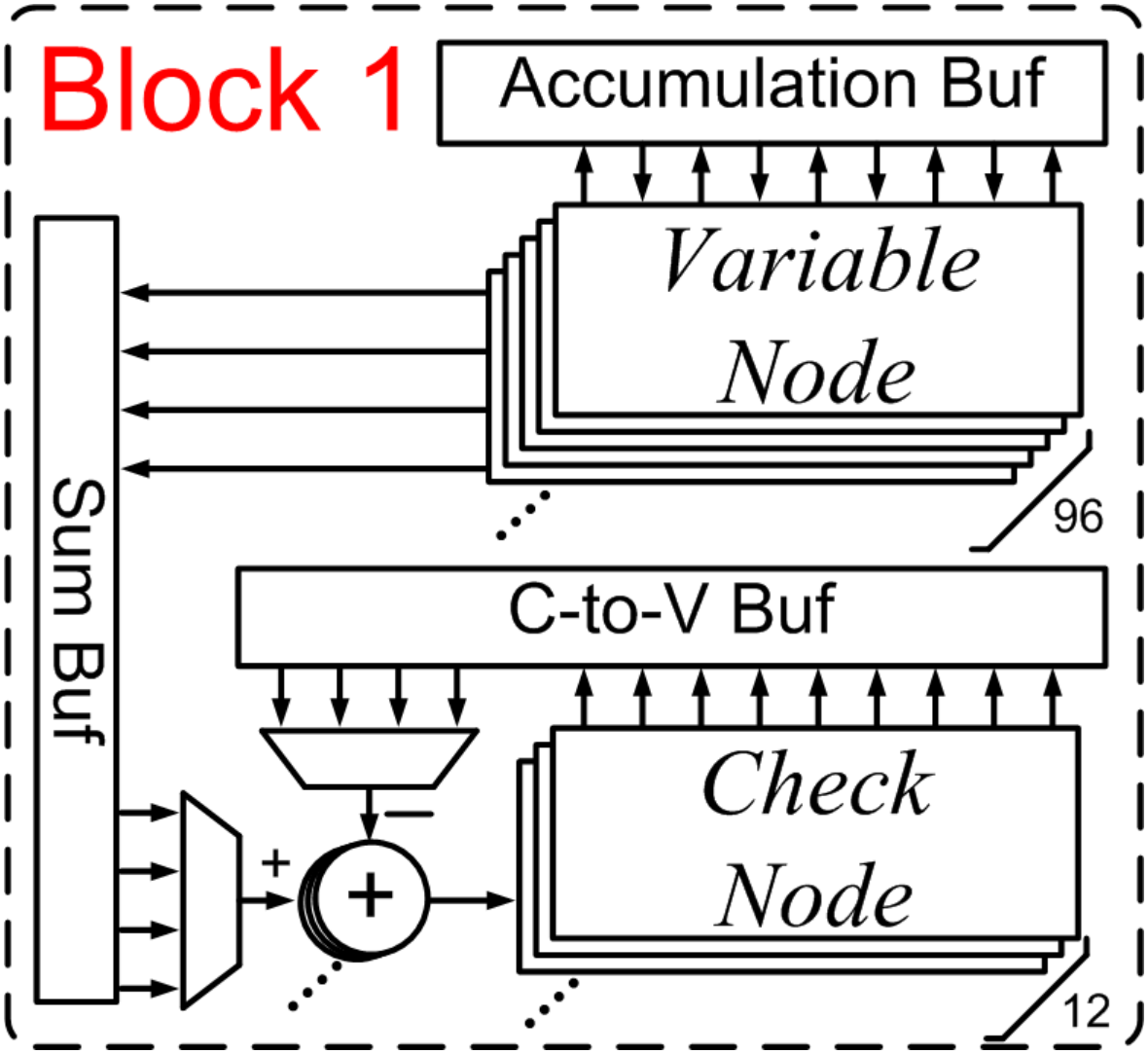
# Matrix Partitioning & Block Design

- Modified H-Matrix: Partitioned into 4 blocks
  - Reducing global communication
  - Accommodating deeply-pipelined architecture



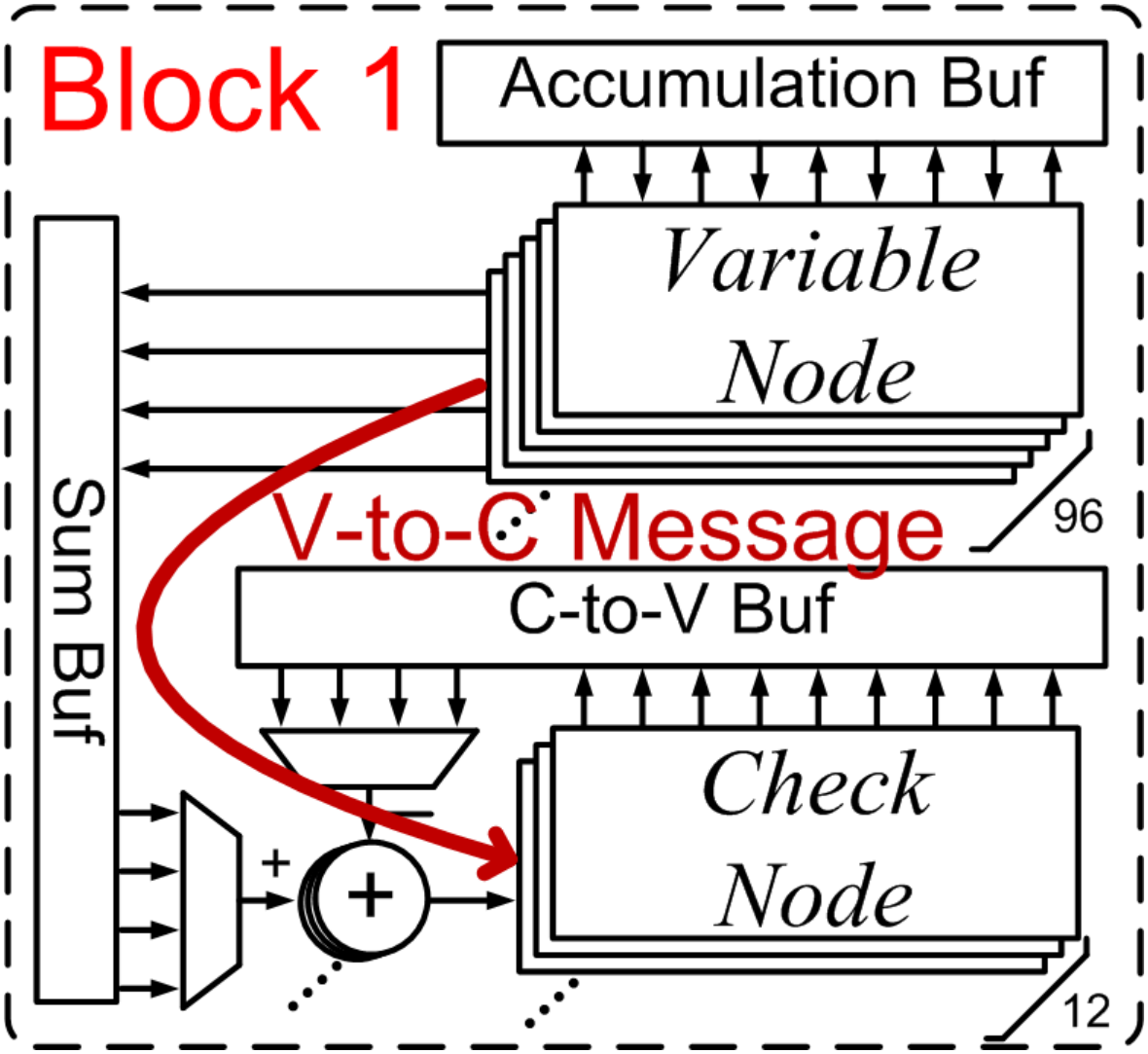
# Block Architecture

- Block Diagram of Block 1
  - 96 Variable Nodes (*VN*)
  - 12 Check Nodes (*CN*)



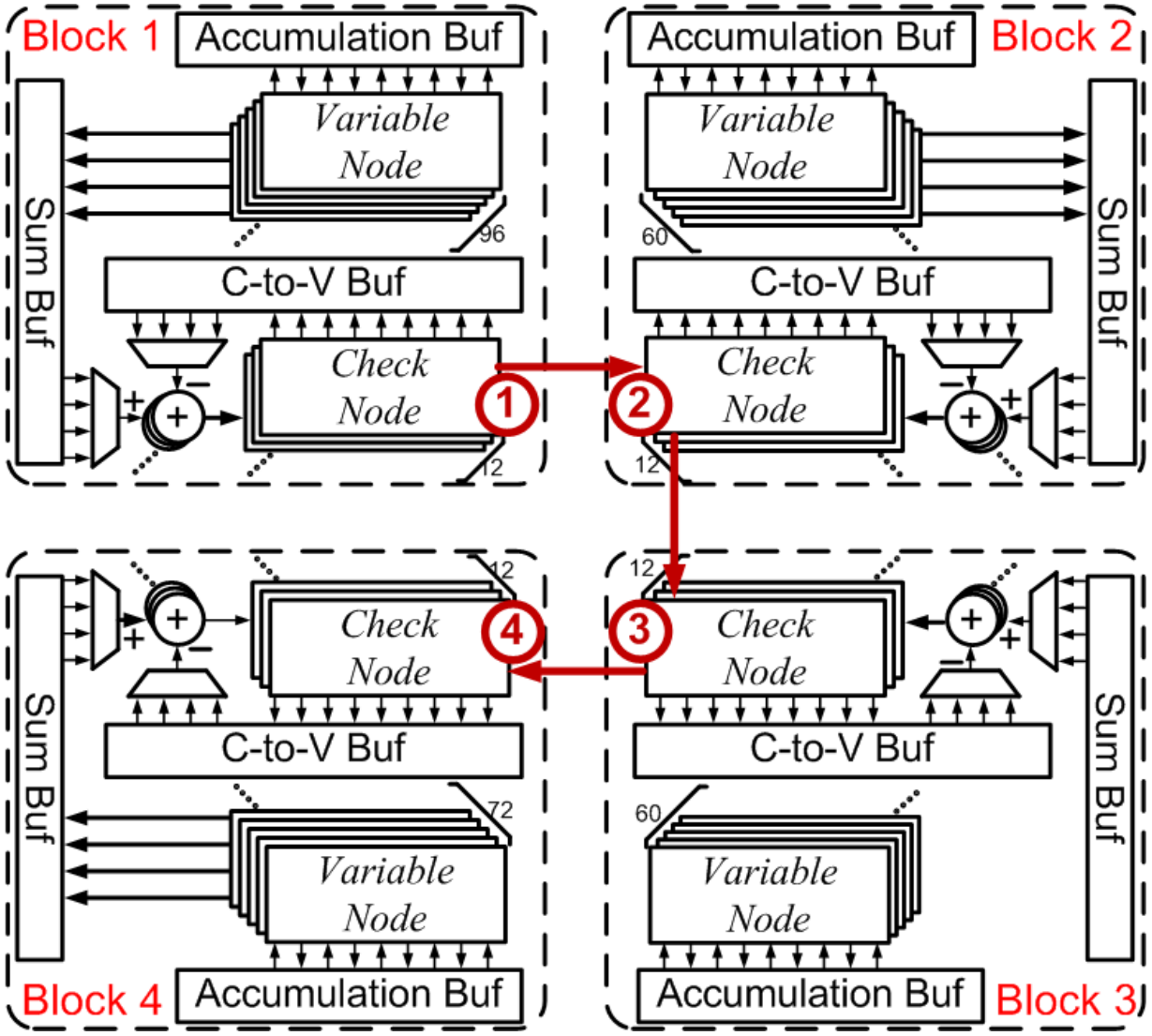
# Block Architecture

- Operation begins from *VN* passing initial V-to-C Message to *CN*



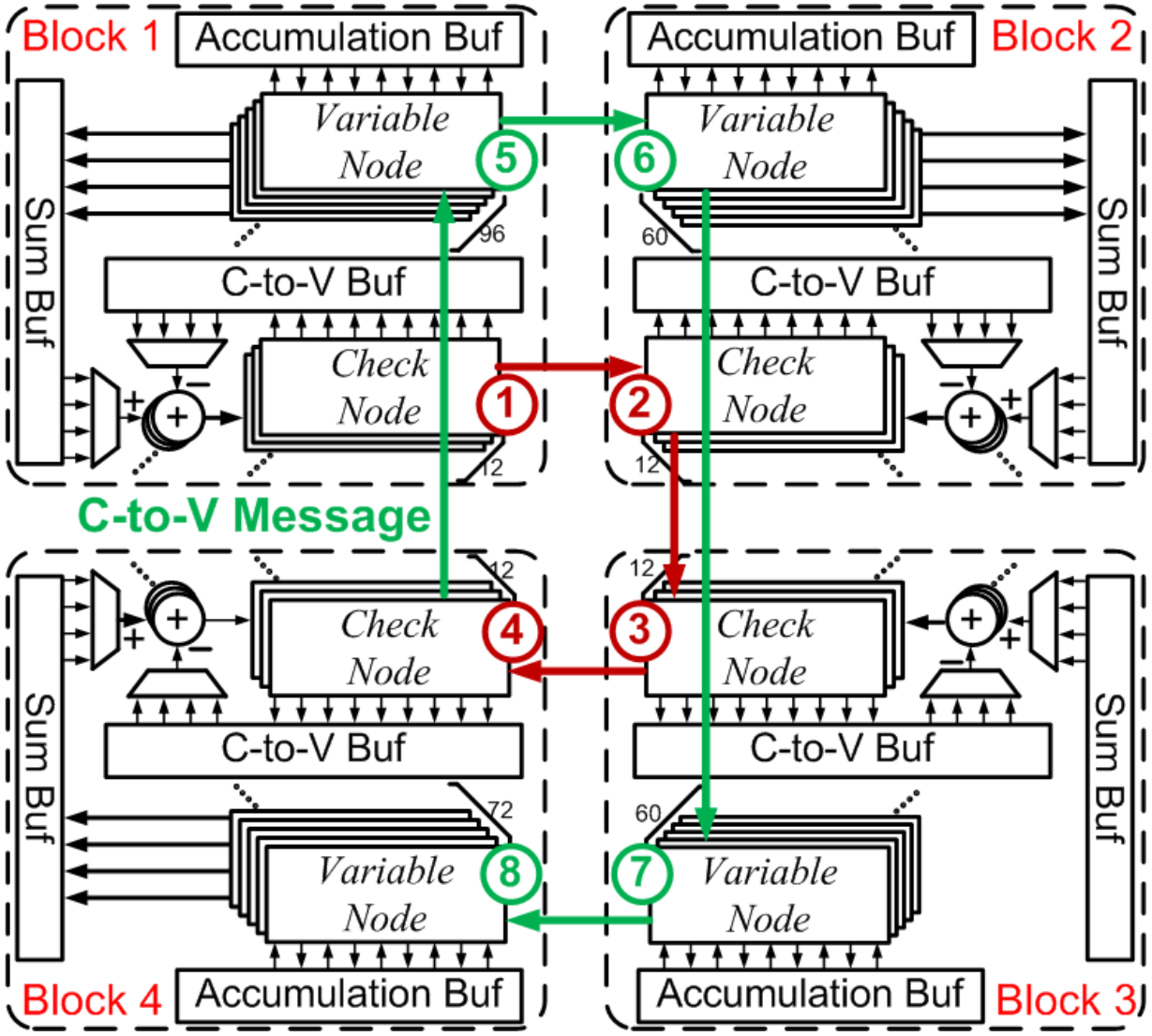
# Relaying Global Messages

- *CN* operation begins in Block 1
- Results are relayed to Blocks 2, 3, and 4 in order



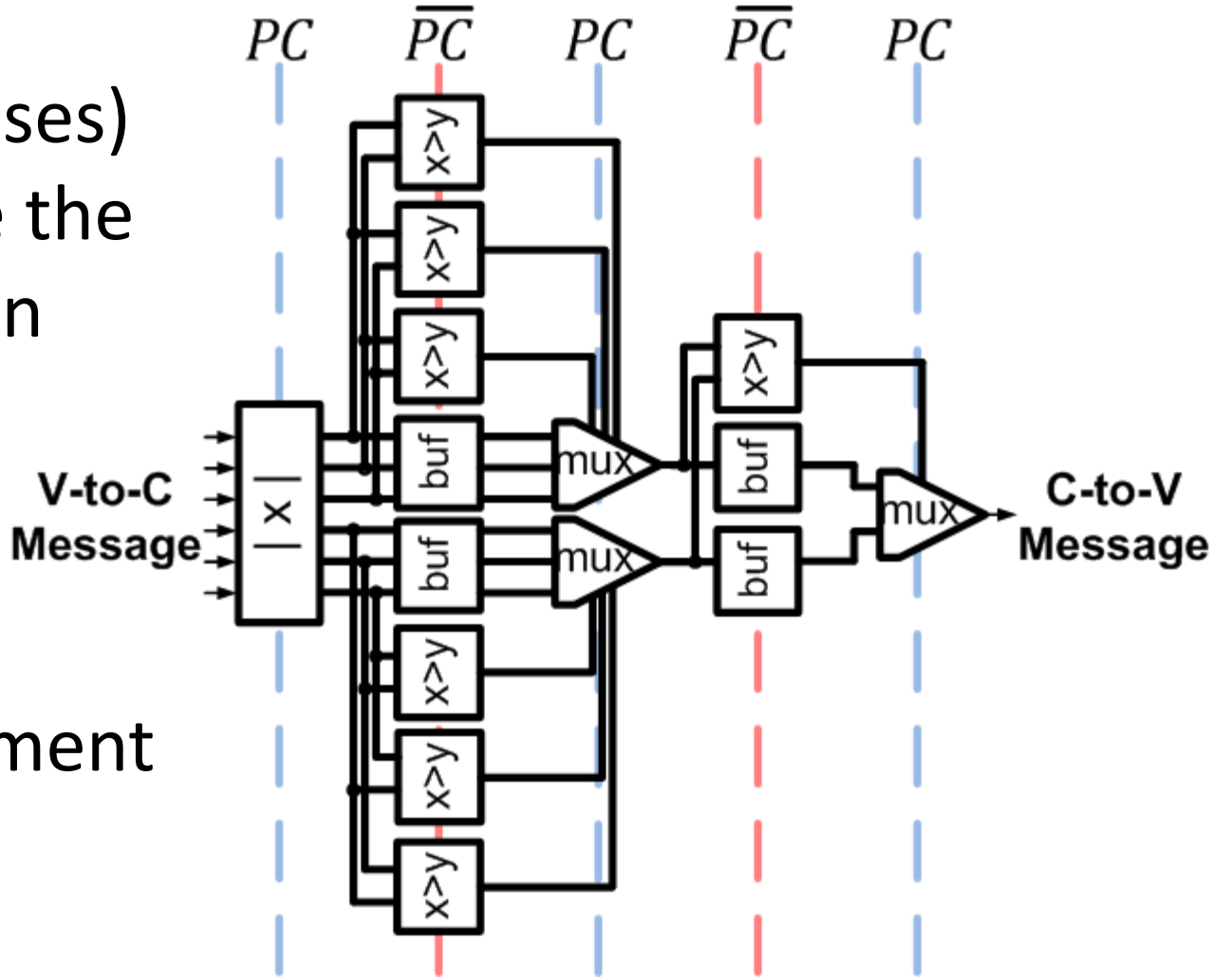
# Interleaving *CN* & *VN*

- *VN* operation follows the *CN* operation
- 24 cycles for one complete decoding iteration
- Four streams computed in parallel



# *CN* Operation Block Diagram

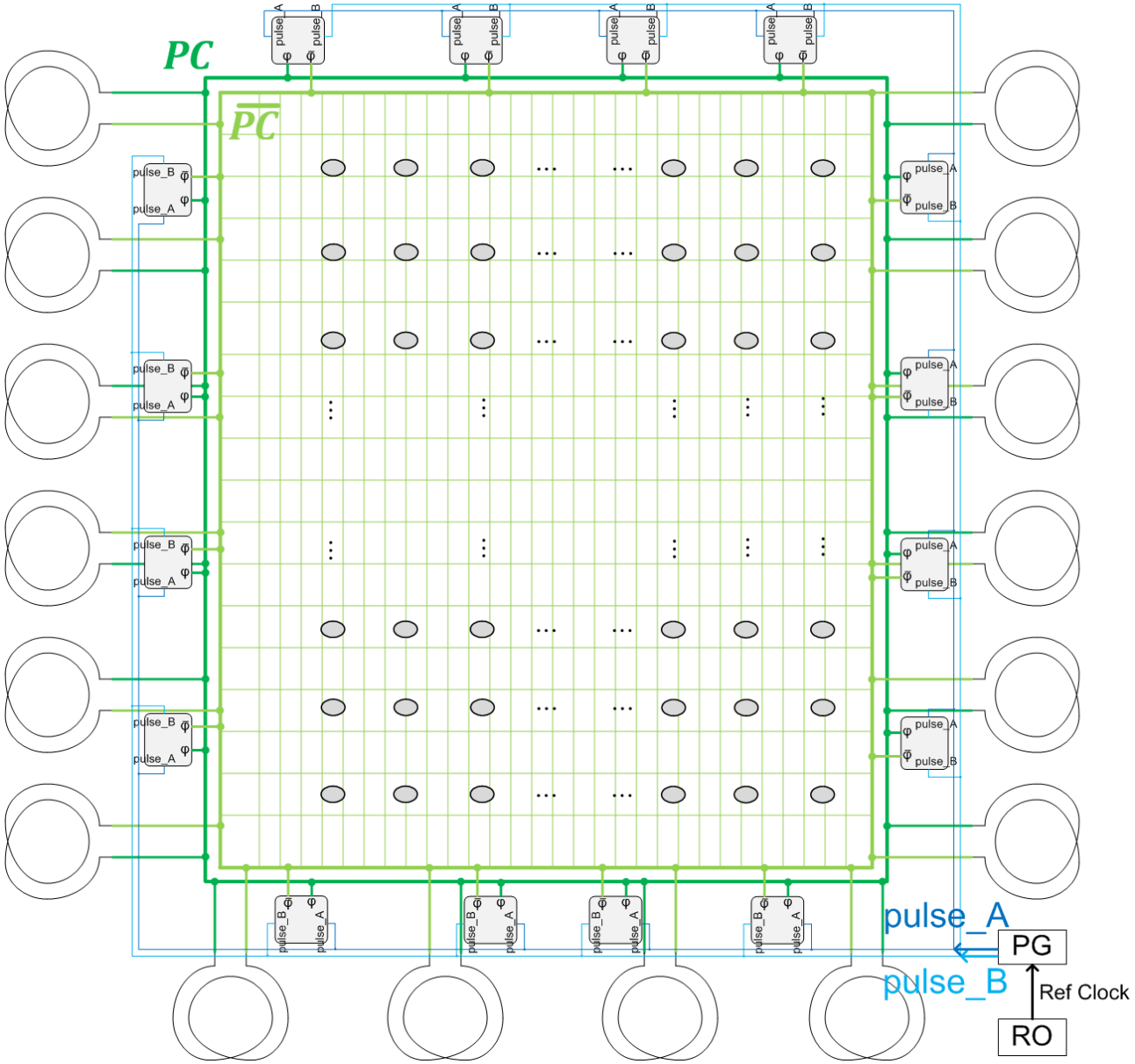
- 2.5 cycles (5 clock phases) to complete the *CN* operation of a block
- Buffers inserted for phase alignment



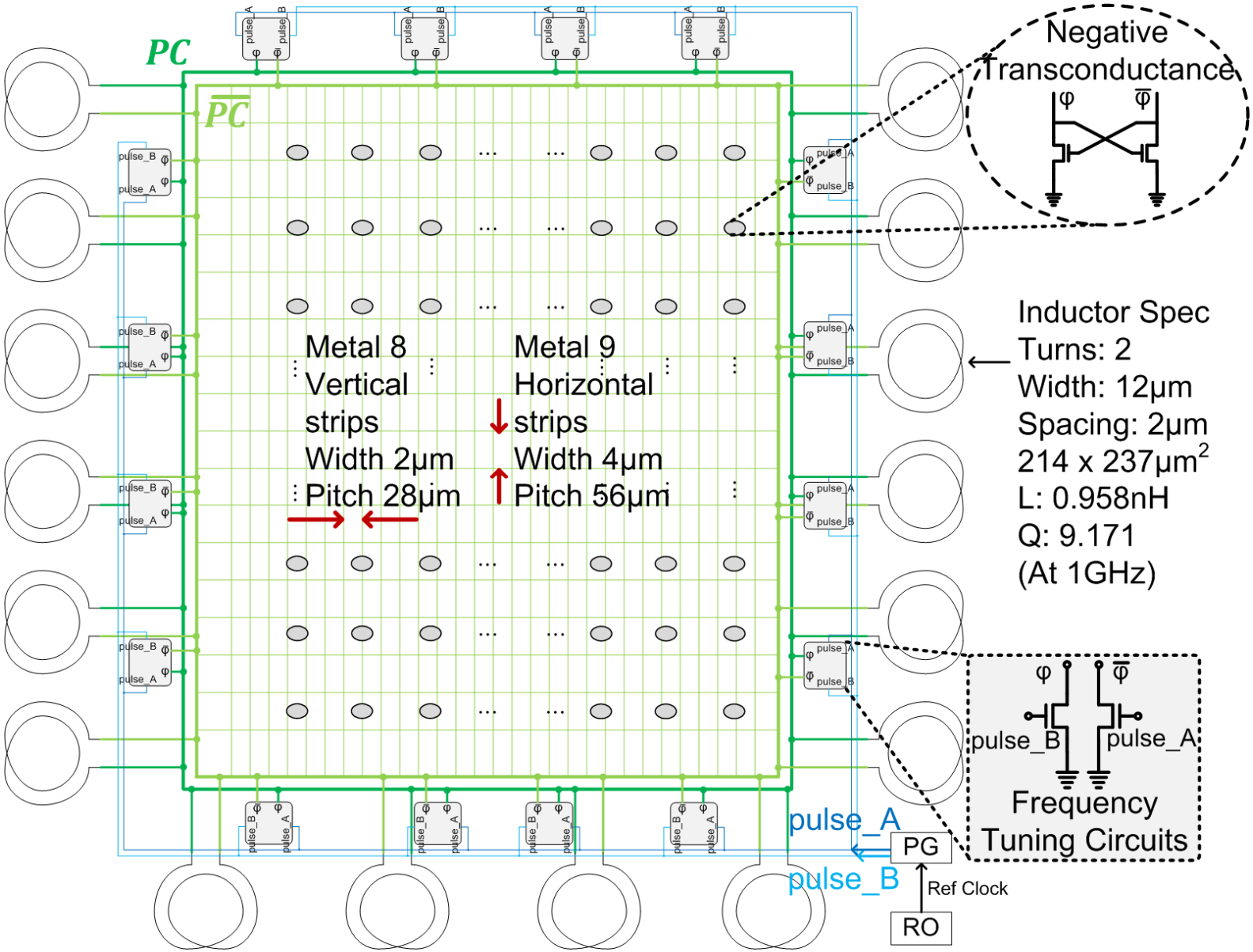
# Outline

- Motivation and Previous Work
- Charge-Recovery Logic
- LDPC Decoder Design and Architecture
- **Power-Clock Distribution Network**
- Design Methodology
- Test Chip Results
- Conclusion

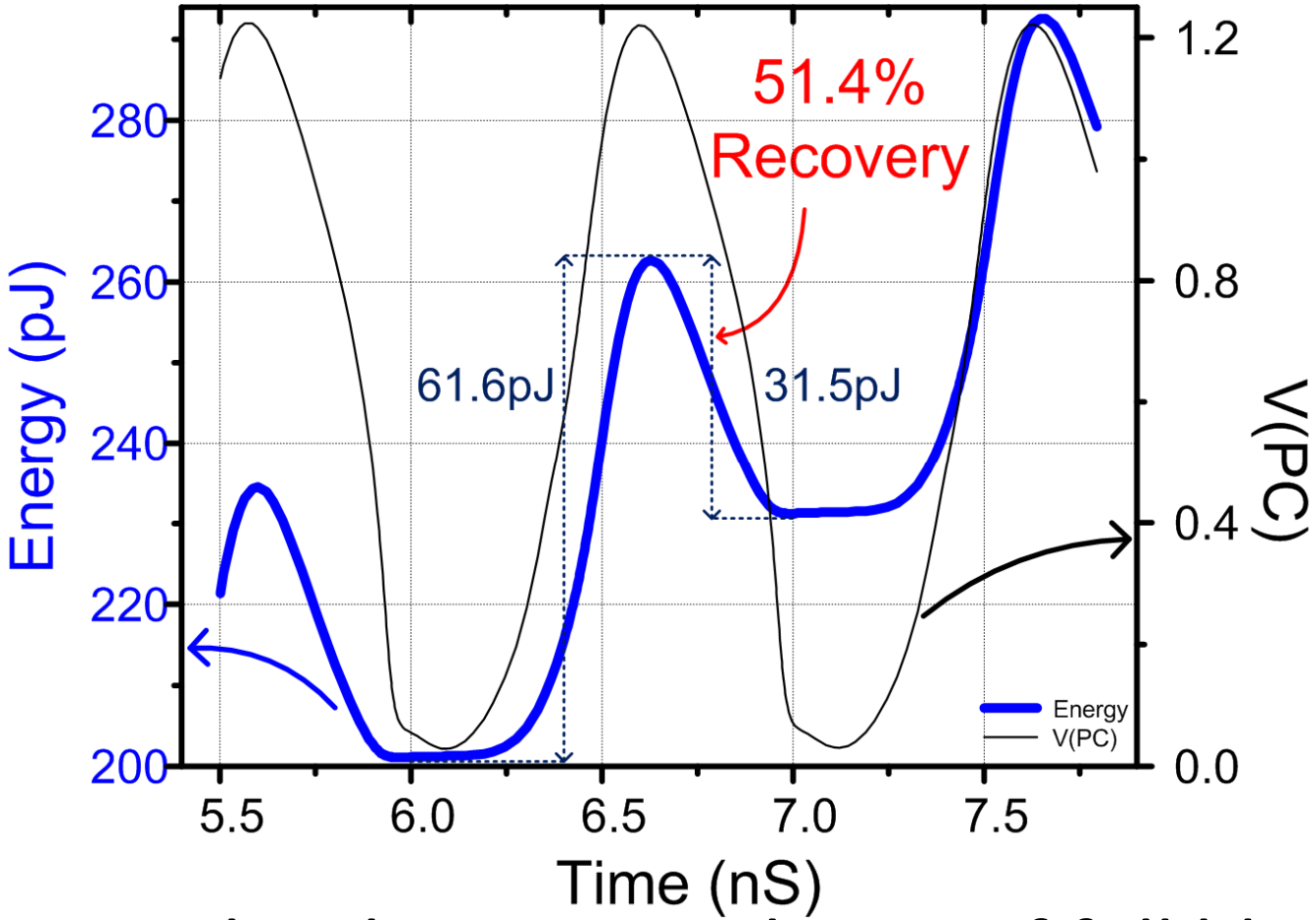
# PC Distribution Network



# PC Distribution Network



# PC Energy Consumption




- Device-level Spice simulation of full block for energy supplied to PC network through one of the sixteen inductors

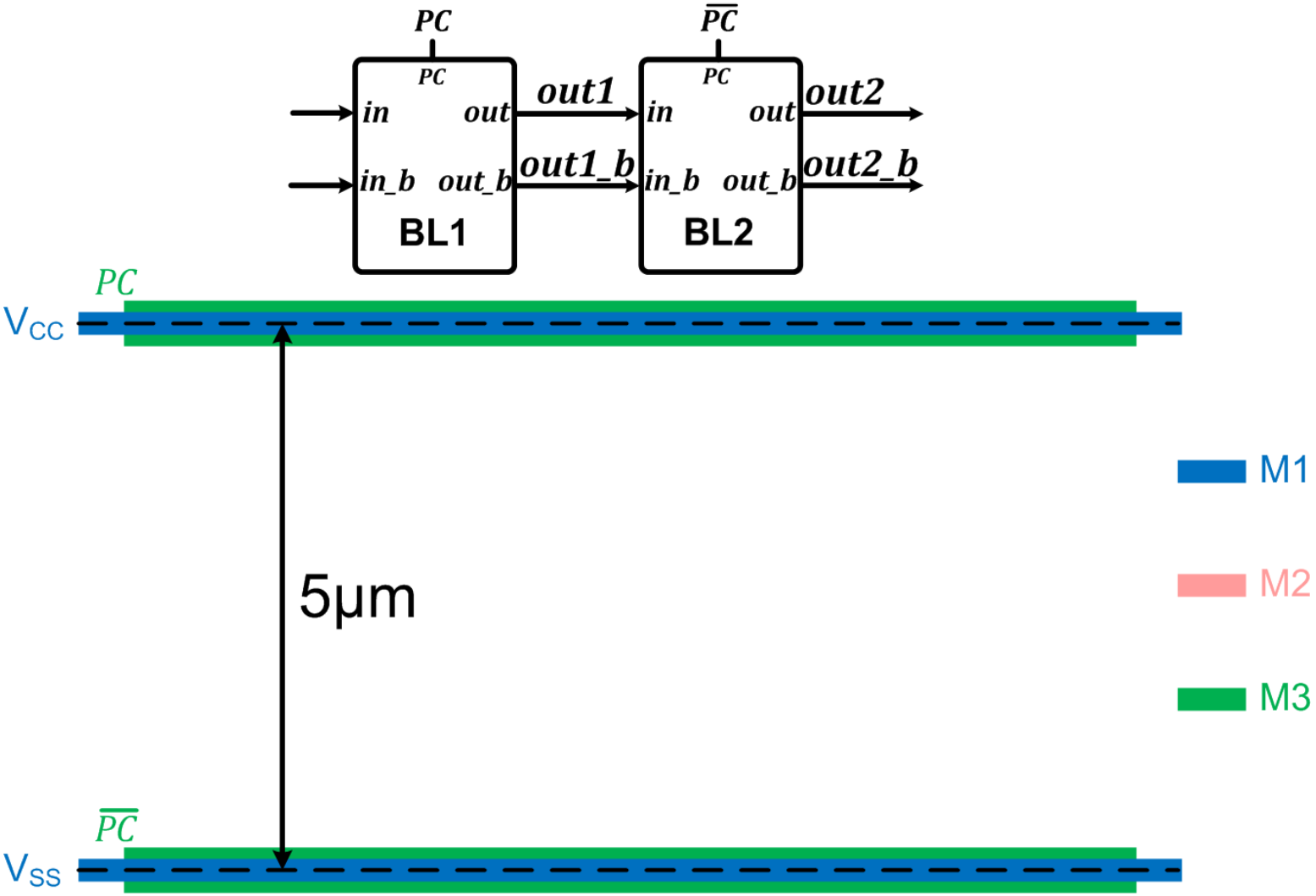
# Outline

- Motivation and Previous Work
- Charge-Recovery Logic
- LDPC Decoder Design and Architecture
- Power-Clock Distribution Network
- **Design Methodology**
- Test Chip Results
- Conclusion

# Design Methodology

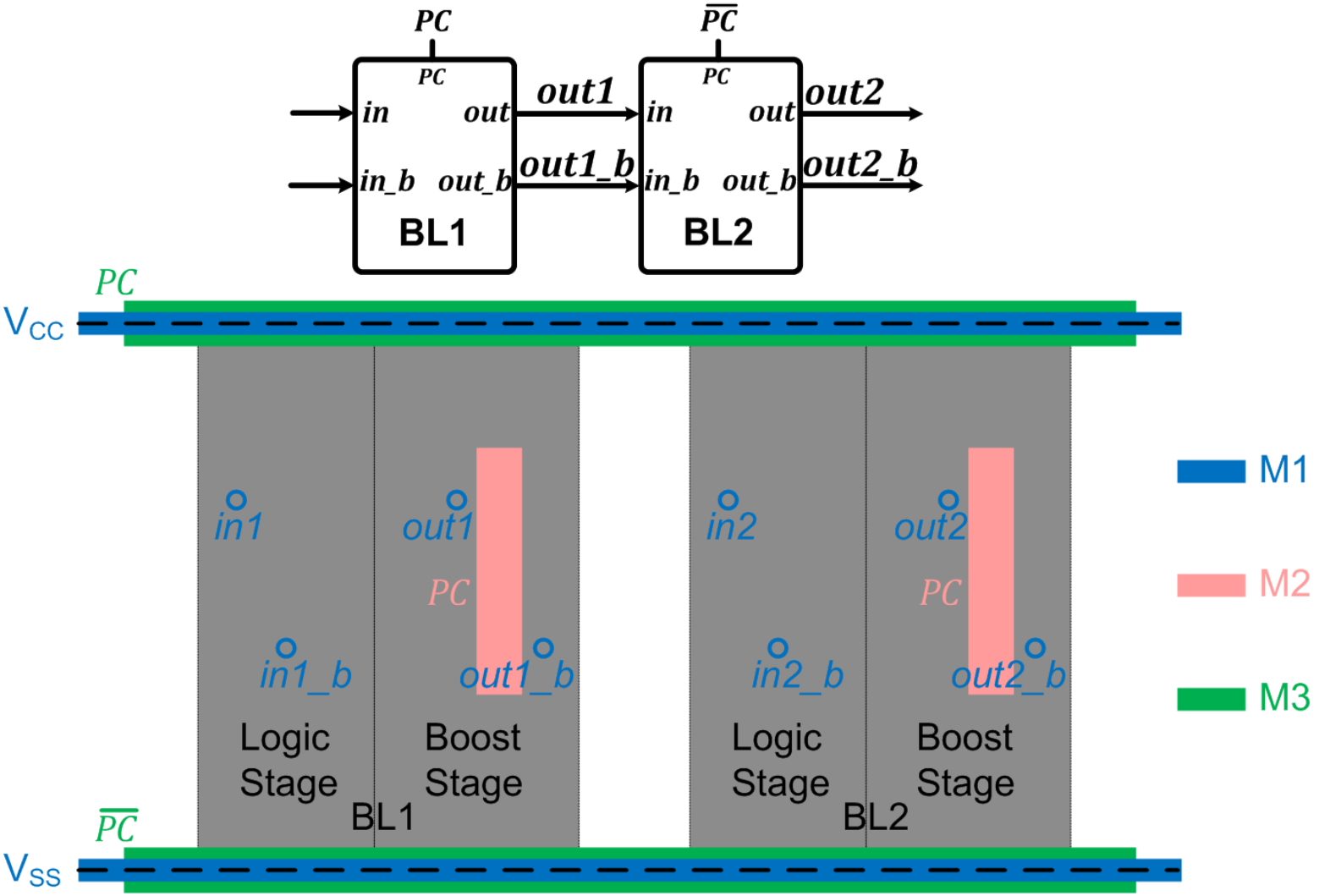
- Semi-custom design flow
  - Created a library of BL gates
    - 52 functions
    - 2-6 different driving strengths per function
    - Up to 25 transistors in each PDN
  - Used commercial EDA tools for:
    - Floorplanning
    - Placement
    - Routing

# Placement & Routing Flow (1/4)



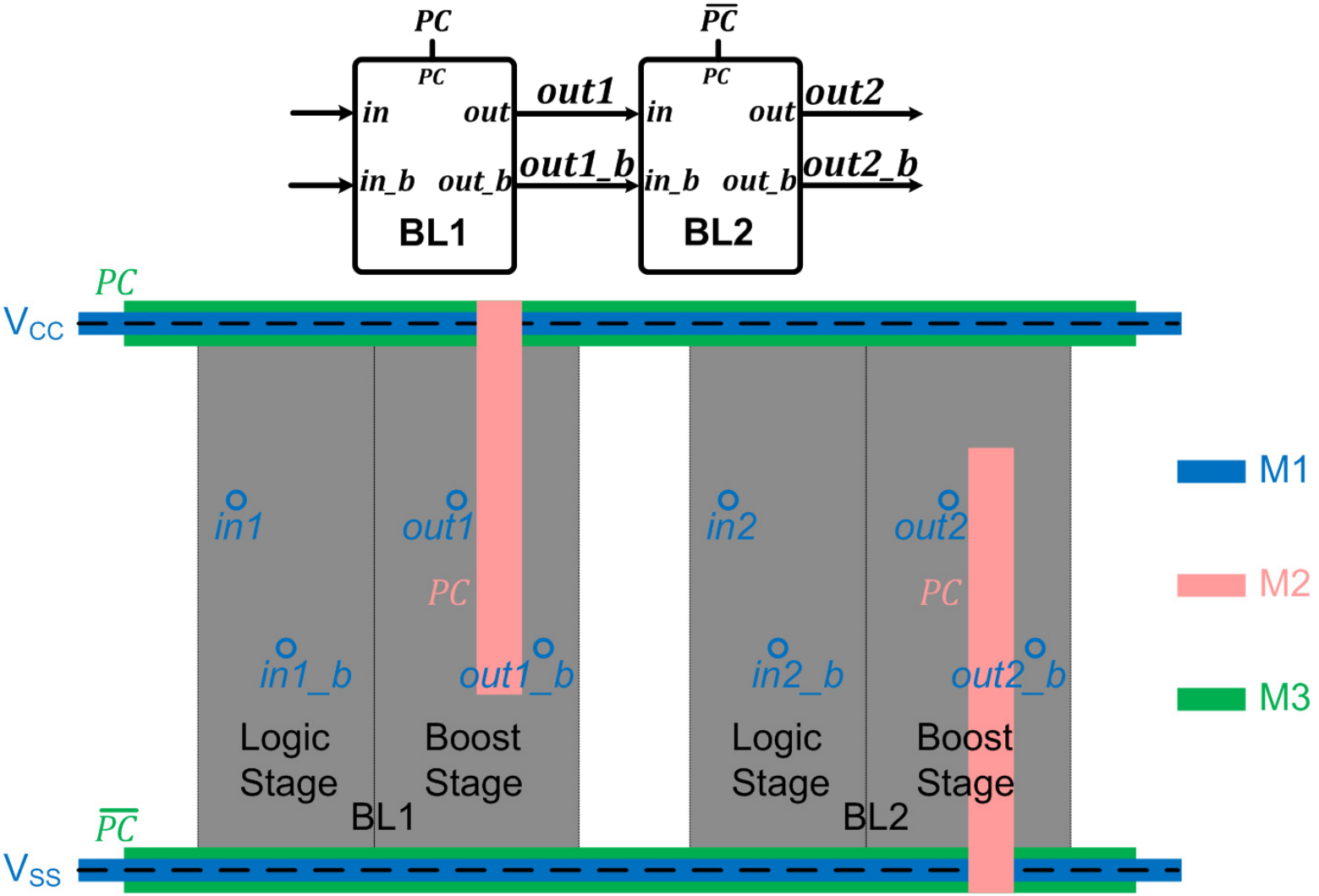

- Create supply grid and PC / $\overline{PC}$ mesh

# Placement & Routing Flow (2/4)



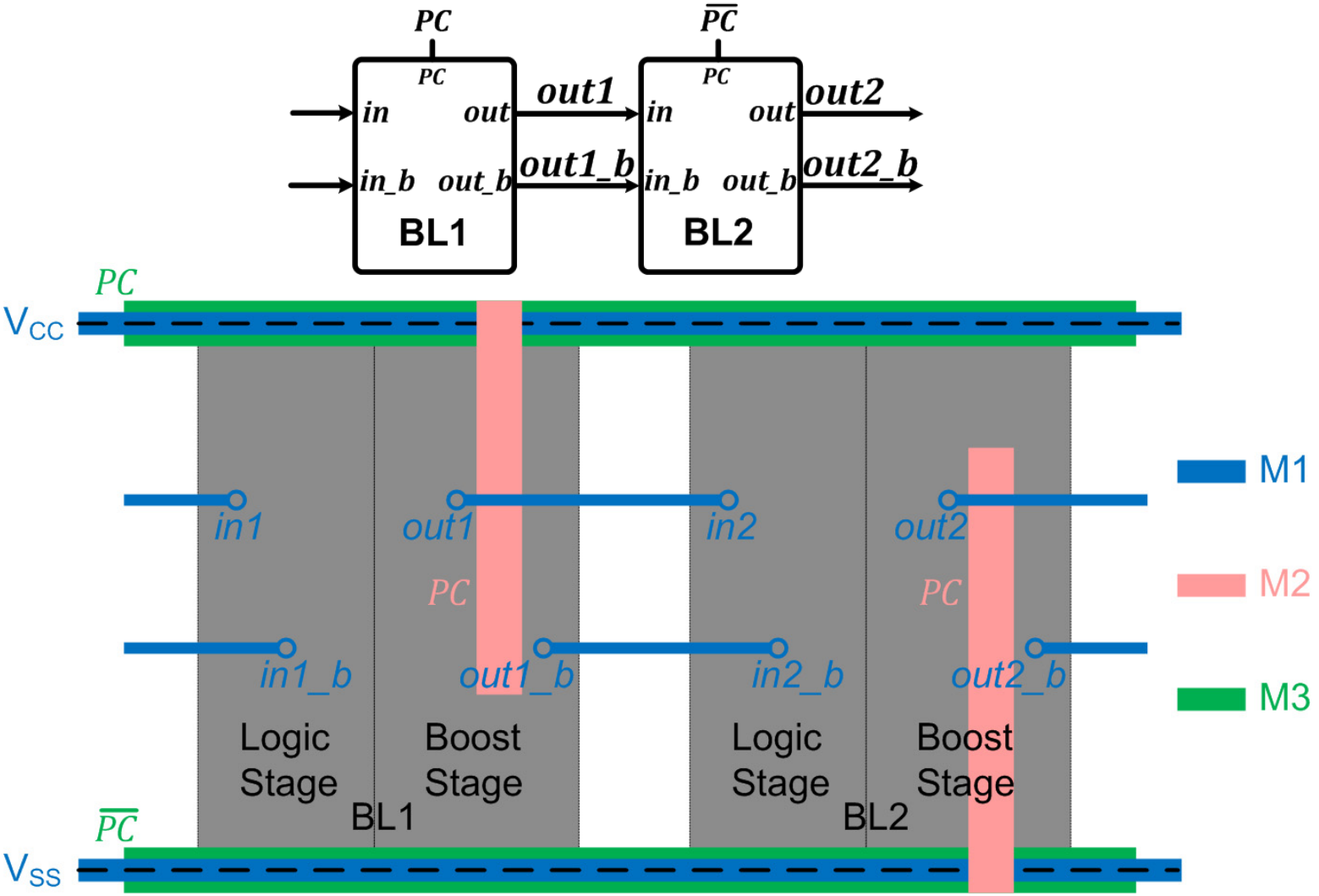

- Place BL cells, trial route, optimize design

# Placement & Routing Flow (3/4)



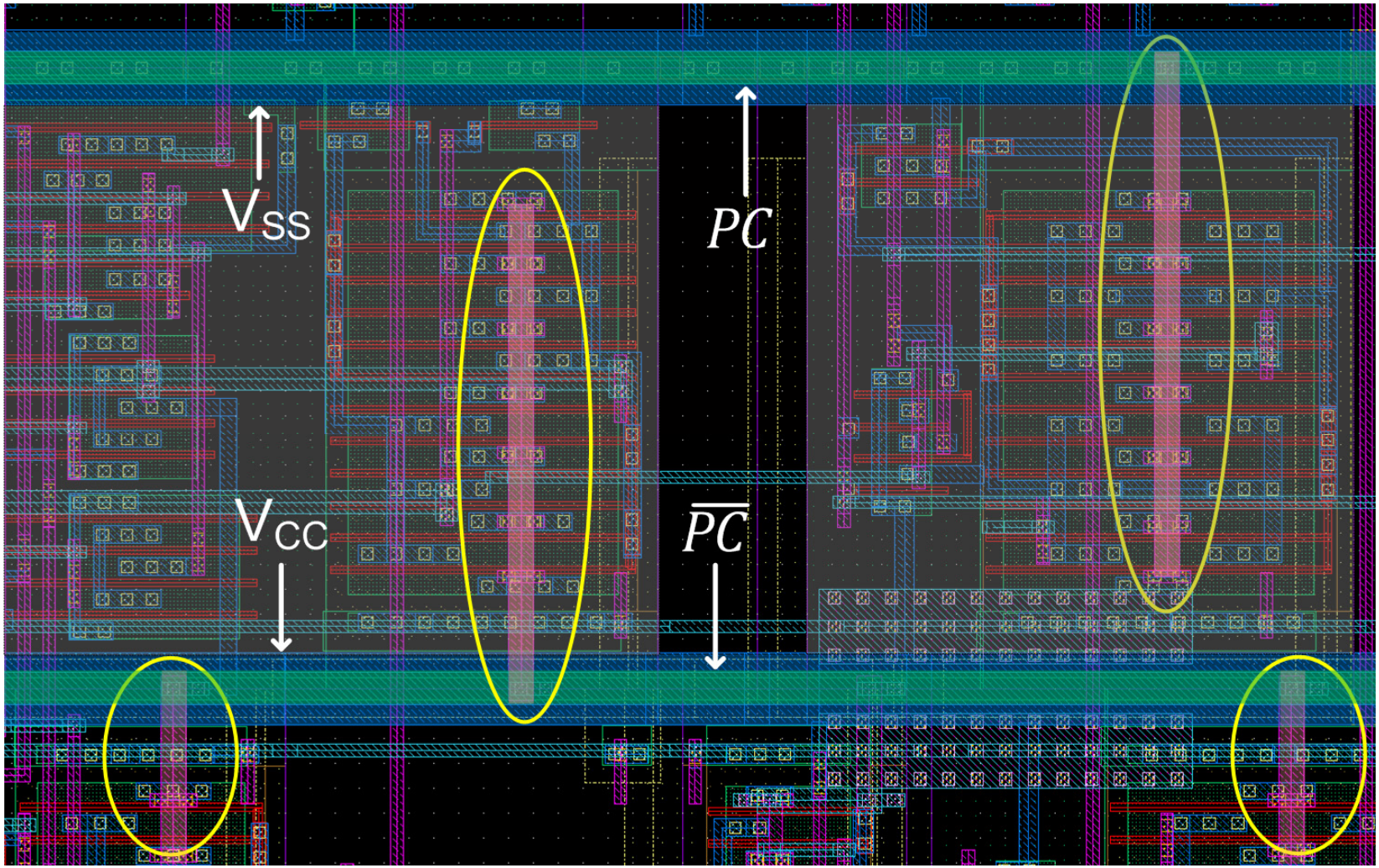

- Route PC and $\overline{PC}$

# Placement & Routing Flow (4/4)




- Route signals

# Actual Layout View

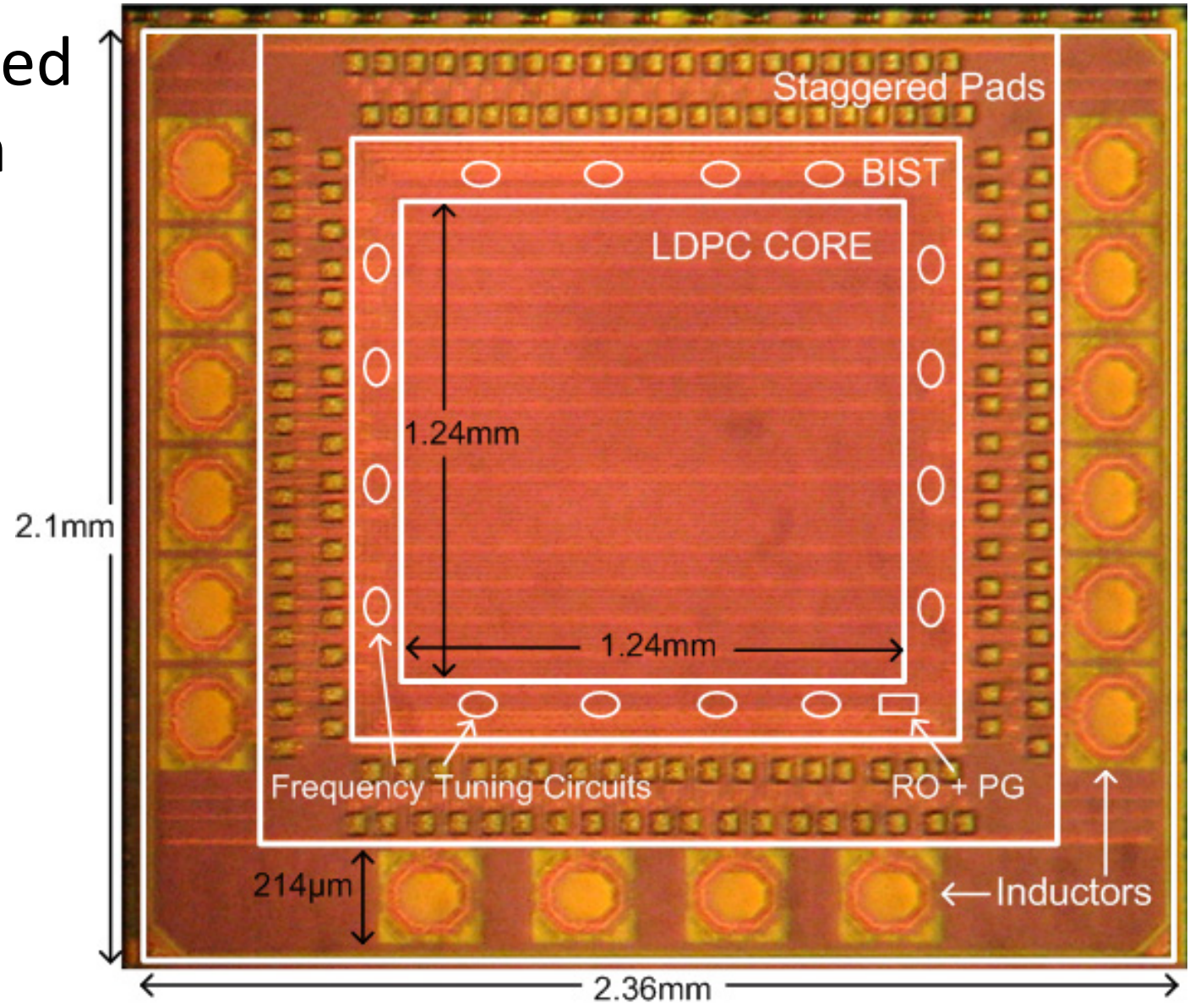


# Outline

- Motivation and Previous Work
- Charge-Recovery Logic
- LDPC Decoder Design and Architecture
- Power-Clock Distribution Network
- Design Methodology
- **Test Chip Results**
- Conclusion

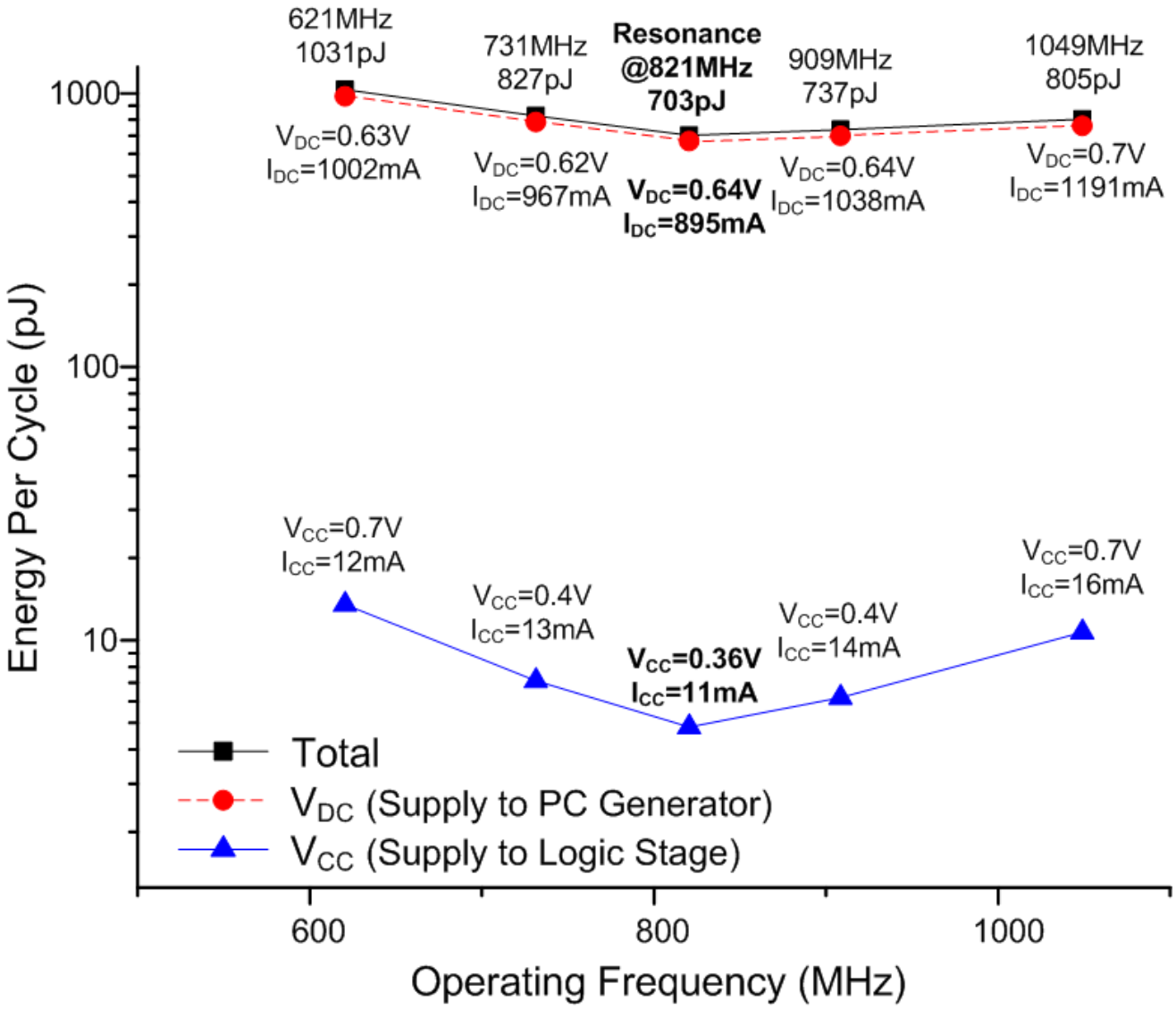
# Chip Microphotograph

- Fabricated in 65nm CMOS



# Energy Measurements

- Minimum energy when self resonating at 821MHz



# Chip Summary and Comparison

| | JSSC '11 | ASSCC '11 | JSSC '12 | ISSCC '13 | This Work |
|---|---|---|---|---|---|
| Technology | 0.13µm | 65nm | 65nm | 65nm | 65nm |
| Code Length | 576~2304 | 576~2304 | 672 | 32 | 576 |
| Core Area ($mm^2$) | 3.03 | 3.36 | 1.56 | 0.063 | 1.54 (2.34 w/ inductors) |
| Frequency (MHz) | 214 | 110 | 197 | 240 | 821 |
| Iterations | 10 | 10 | 5 | 5 | 10 |
| Throughput (Gbps) | 0.847 | 1.056 | 5.79 | 0.38 | 7.882 |
| Power (mW) | 342 | 115 | 361 | 4 | 576.8 |
| Energy Efficiency (pJ/bit/iteration) | 13.54 [1] | 21.80 | 12.48 [2, 3] | 2.08 [3] | 7.32 |
| Area Efficiency ($Gbps/mm^2$) | 2.24 [1] | 0.31 | 2.12 [2, 3] | 3.05 [3] | 5.12 (3.35 w/ inductors) |

[1]Scaled to 65nm, 1.0V supply [2]Scaled to data throughput [3]Normalized to 10 iterations

# Chip Summary and Comparison

| | JSSC '11 | ASSCC '11 | JSSC '12 | ISSCC '13 | This Work | |
|---|---|---|---|---|---|---|
| Technology | 0.13µm | 65nm | 65nm | 65nm | 65nm | |
| Code Length | 576~2304 | 576~2304 | 672 | 32 | 576 | |
| Core Area ($mm^2$) | 3.03 | 3.36 | 1.56 | 0.063 | 1.54 (2.34 w/ inductors) | |
| Frequency (MHz) | 214 | 110 | 197 | 240 | 821 | |
| Iterations | 10 | 10 | 5 | 5 | 10 | |
| Throughput (Gbps) | 0.847 | 1.056 | 5.79 | 0.38 | 7.882 | |
| Power (mW) | 342 | 115 | 361 | 4 | 576.8 | |
| Energy Efficiency (pJ/bit/iteration) | 13.54 [1] | 21.80 | 12.48 [2, 3] | 2.08 [3] | 7.32 | **1.7×** |
| Area Efficiency ($Gbps/mm^2$) | 2.24 [1] | 0.31 | 2.12 [2, 3] | 3.05 [3] | 5.12 (3.35 w/ inductors) | **2.3×** |

[1]Scaled to 65nm, 1.0V supply [2]Scaled to data throughput [3]Normalized to 10 iterations

# Conclusion

- 576-bit charge-recovery LDPC decoder
- Self-resonates at 821MHz, functional up to 1.05GHz
- Semi-custom design methodology
- 1.7× better energy efficiency
- 2.3× better area efficiency
- Demonstrated potential of charge-recovery logic to large-scale DSP applications

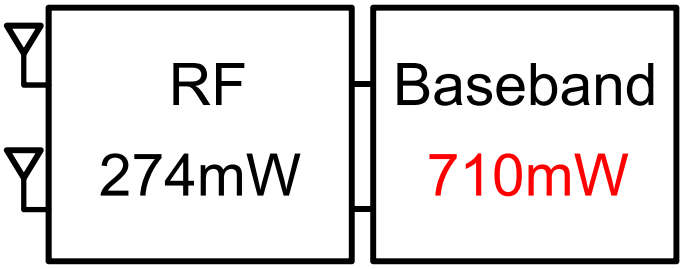
# A Scalable 1.5-to-6Gb/s 6.2-to-38.1mW LDPC Decoder for 60GHz Wireless Networks in 28nm UTBB FDSOI

Matthew Weiner[1], Milovan Blagojevic[1,2,3], Sergey Skotnikov[4], Andreas Burg[4], Philippe Flatresse[3], Borivoje Nikolic[1]

[1]University of California, Berkeley, CA
[2]Institute Supérieur d'Electronique de Paris, Paris, France
[3]STMicroelectronics, Crolles, France
[4]EPFL, Lausanne, Switzerland

# 60GHz wireless networks

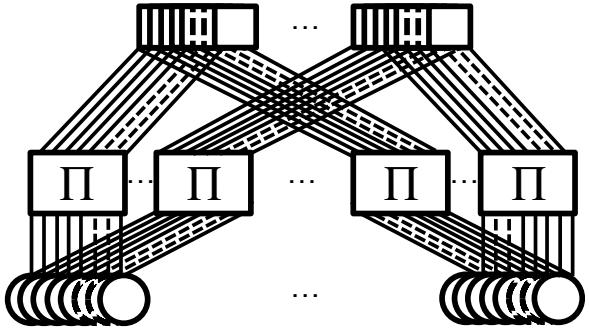
**Uses:** high-speed WiFi or device-to-device communication




Saito et al., JSSC, Dec. 2013

**Problem:** excessive power in digital baseband (FEC)

**Goal:** reduce power of all modes of LDPC decoder to lower total power



# Outline

- LDPC decoding background
- IEEE 802.11ad/WiGig standard
- Decoder architecture
- Implementation and results

# LDPC codes and hardware

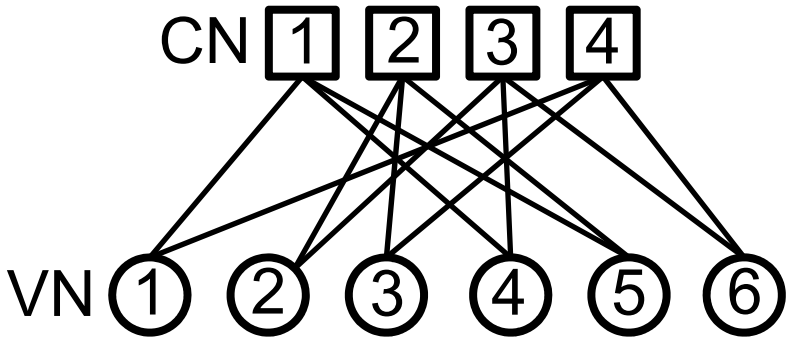
## Parity check matrix

| | | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 0 | 0 | 1 | 1 | 0 |
| CN | 0 | 1 | 1 | 0 | 1 | 0 |
| 1-4 | 0 | 1 | 0 | 1 | 0 | 1 |
| | 1 | 0 | 1 | 0 | 0 | 1 |

VN 1-6

⇔

## Tanner graph

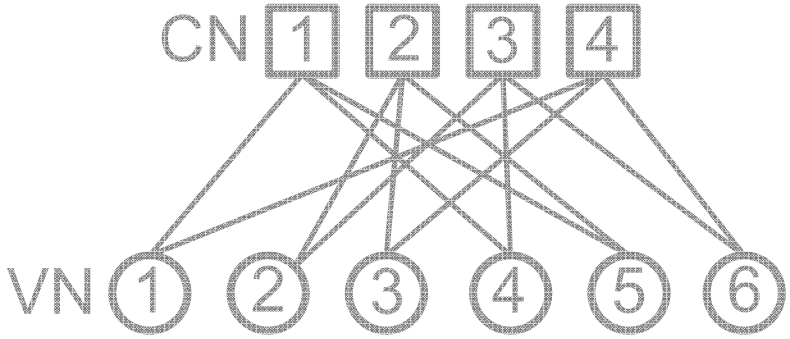


# LDPC codes and hardware

## Parity check matrix

| CN 1-4 \ VN 1-6 | | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 0 | 0 | 1 | 1 | 0 |
| | 0 | 1 | 1 | 0 | 1 | 0 |
| | 0 | 1 | 0 | 1 | 0 | 1 |
| | 1 | 0 | 1 | 0 | 0 | 1 |

## Tanner graph

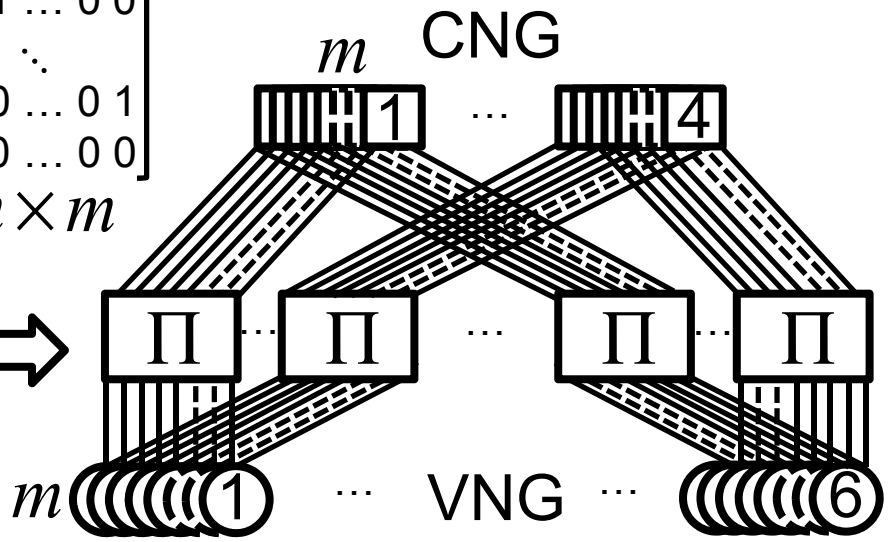



$$\begin{bmatrix} 0\,1\,0\,\ldots\,0\,0 \\ 0\,0\,1\,\ldots\,0\,0 \\ \ddots \\ 0\,0\,0\,\ldots\,0\,1 \\ 1\,0\,0\,\ldots\,0\,0 \end{bmatrix}$$

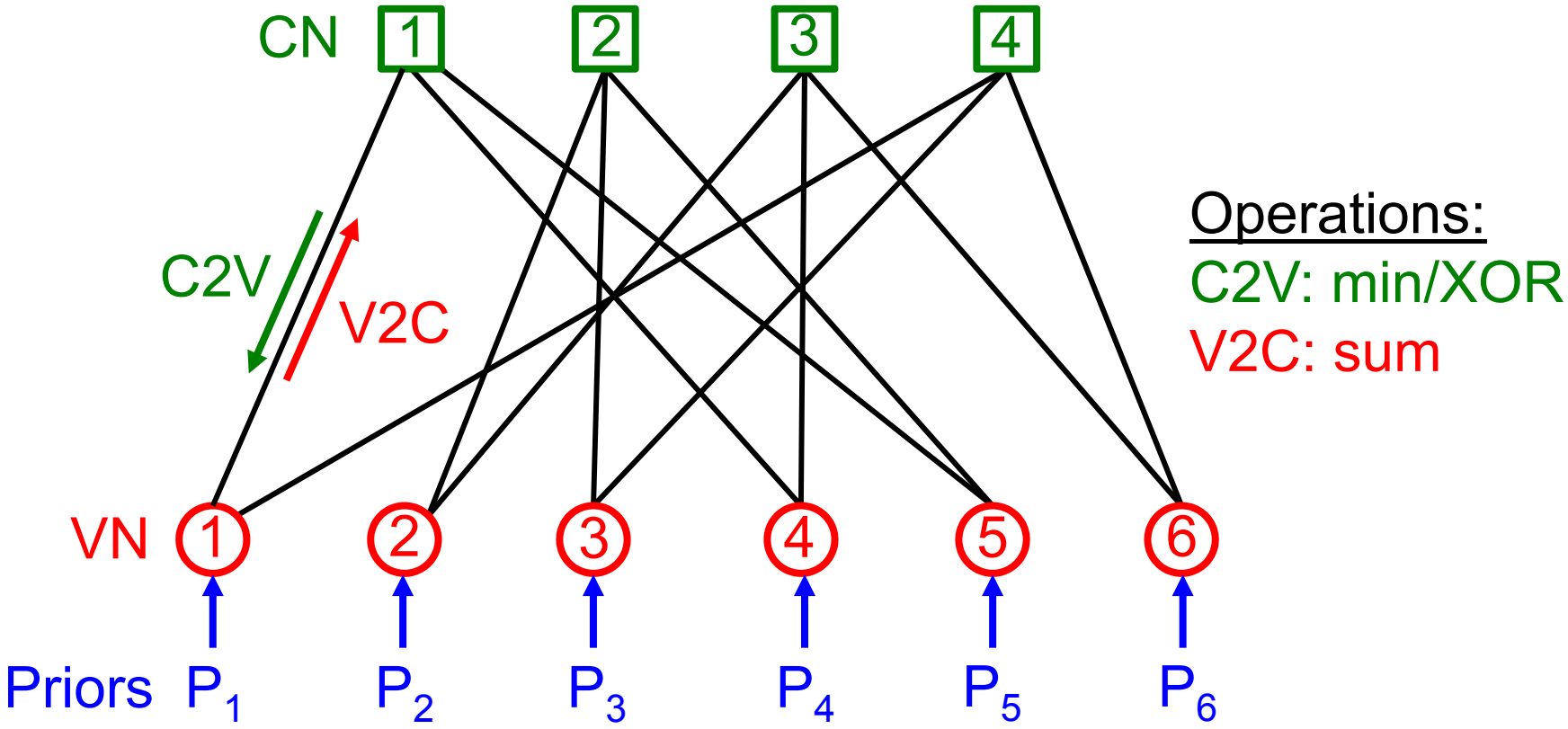
$m \times m$

VNG 1-6

| CNG 1-4 | | | | | | |
|---|---|---|---|---|---|---|
| | 20 | 31 | 0 | 5 | | |
| | 2 | 24 | 4 | 17 | 1 | |
| | 6 | | 7 | 22 | 3 | 9 |
| | 1 | 16 | 39 | 41 | 8 | 11 |



# Message passing decoding




"Marginalize" messages to avoid recycling info every iteration

# The IEEE 802.11ad/WiGig standard

672 bits

Rate 3/4

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 35 | 19 | 41 | 22 | 40 | 41 | 39 | 6 | 28 | 18 | 17 | 3 | 28 | | | |
| 29 | 30 | 0 | 8 | 33 | 22 | 17 | 4 | 27 | 28 | 20 | 27 | 24 | 23 | | |
| 37 | 31 | 18 | 23 | 11 | 21 | 6 | 20 | 32 | 9 | 12 | 29 | | 0 | 13 | |
| 25 | 22 | 4 | 34 | 31 | 3 | 14 | 15 | 4 | | 14 | 18 | 13 | 13 | 22 | 24 |

Rate 5/8

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20 | 36 | 34 | 31 | 20 | 7 | 41 | 34 | | 10 | 41 | | | | | |
| 30 | 27 | | 18 | | 12 | 20 | 14 | 2 | 25 | 15 | 6 | | | | |
| 35 | | 41 | | 40 | | 39 | | 28 | | | 3 | 28 | | | |
| 29 | | 0 | | | 22 | | 4 | | 28 | | 27 | 24 | 23 | | |
| | 31 | | 23 | | 21 | | 20 | | 9 | 12 | | | 0 | 13 | |
| | 22 | | 34 | 31 | | 14 | | 4 | | | | | | 22 | 24 |

← Layer

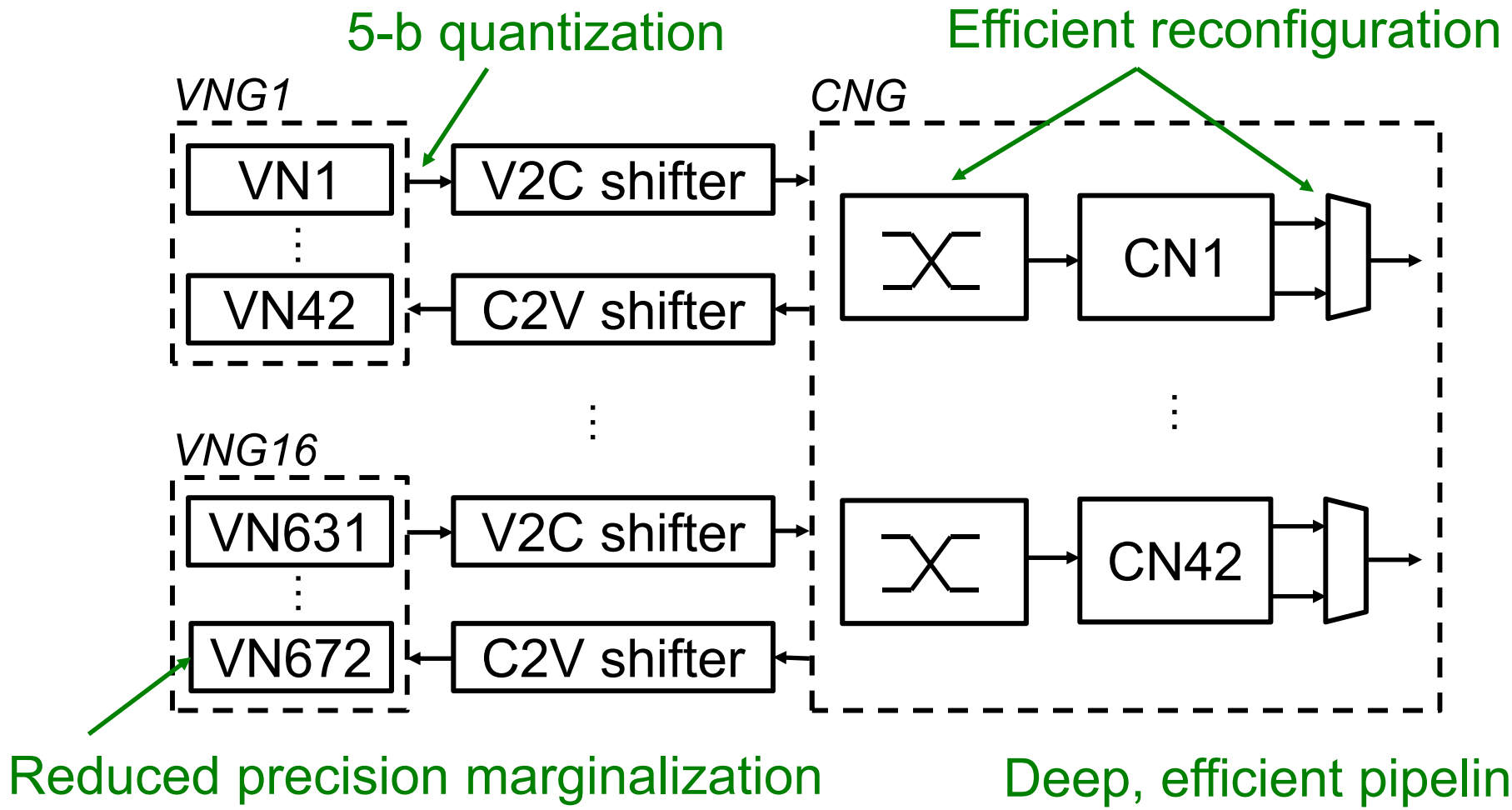
42x42

Rate 1/2

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 40 | | 38 | | 13 | | 5 | | 18 | | | | | | | |
| 34 | | 35 | | 27 | | | 30 | 2 | 1 | | | | | | |
| | 36 | | 31 | | 7 | | 34 | | 10 | 41 | | | | | |
| | 27 | | 18 | | 12 | 20 | | | | 15 | 6 | | | | |
| 35 | | 41 | | 40 | | 39 | | 28 | | | 3 | 28 | | | |
| 29 | | 0 | | | 22 | | 4 | | 28 | | 27 | | 23 | | |
| | 31 | | 23 | | 21 | | 20 | | | 12 | | | 0 | 13 | |
| | 22 | | 34 | 31 | | 14 | | 4 | | | | 13 | | 22 | 24 |

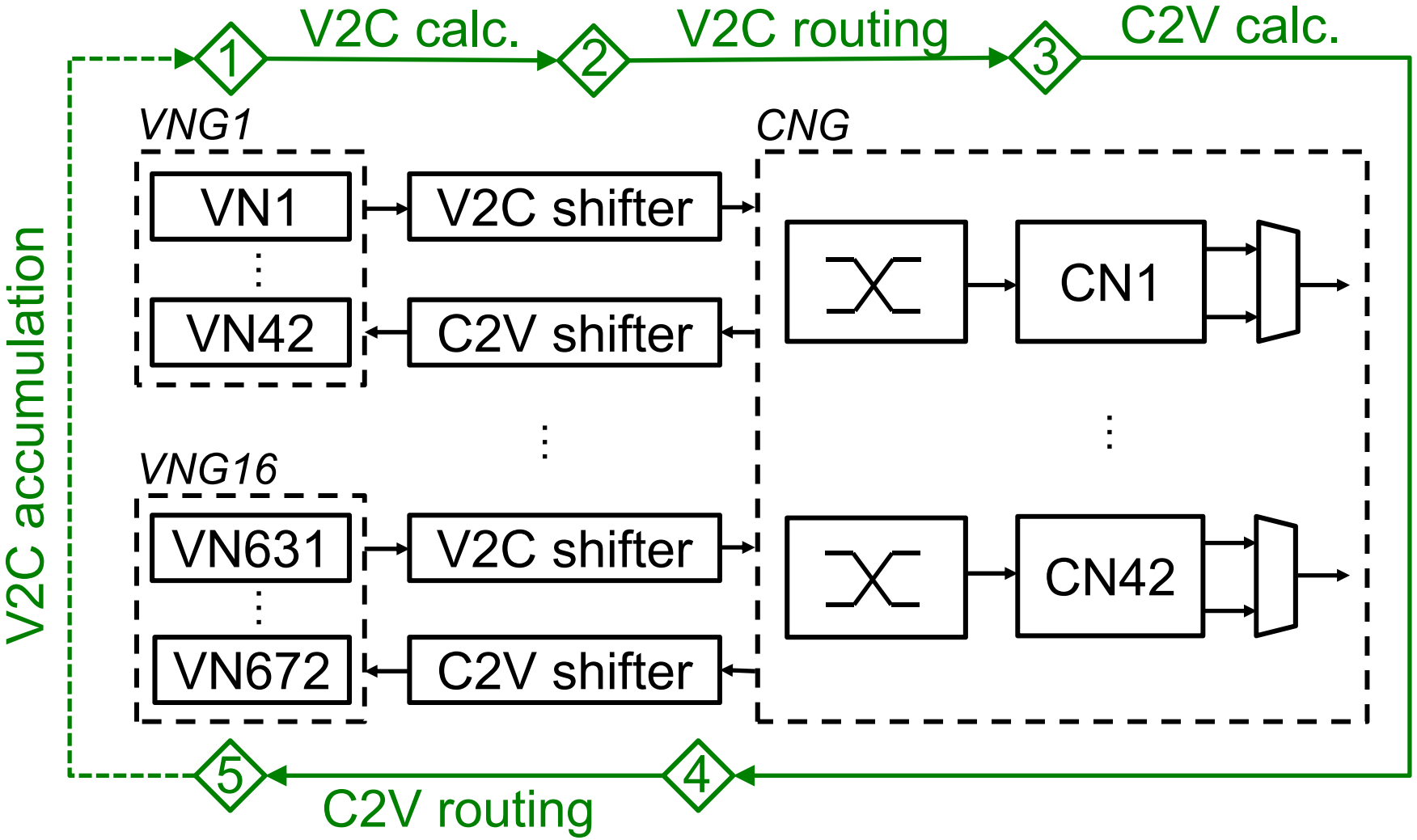
3 throughputs (1.5, 3, 6Gb/s) and 4 rates (13/16, 3/4, 5/8, 1/2)

# Proposed decoder architecture



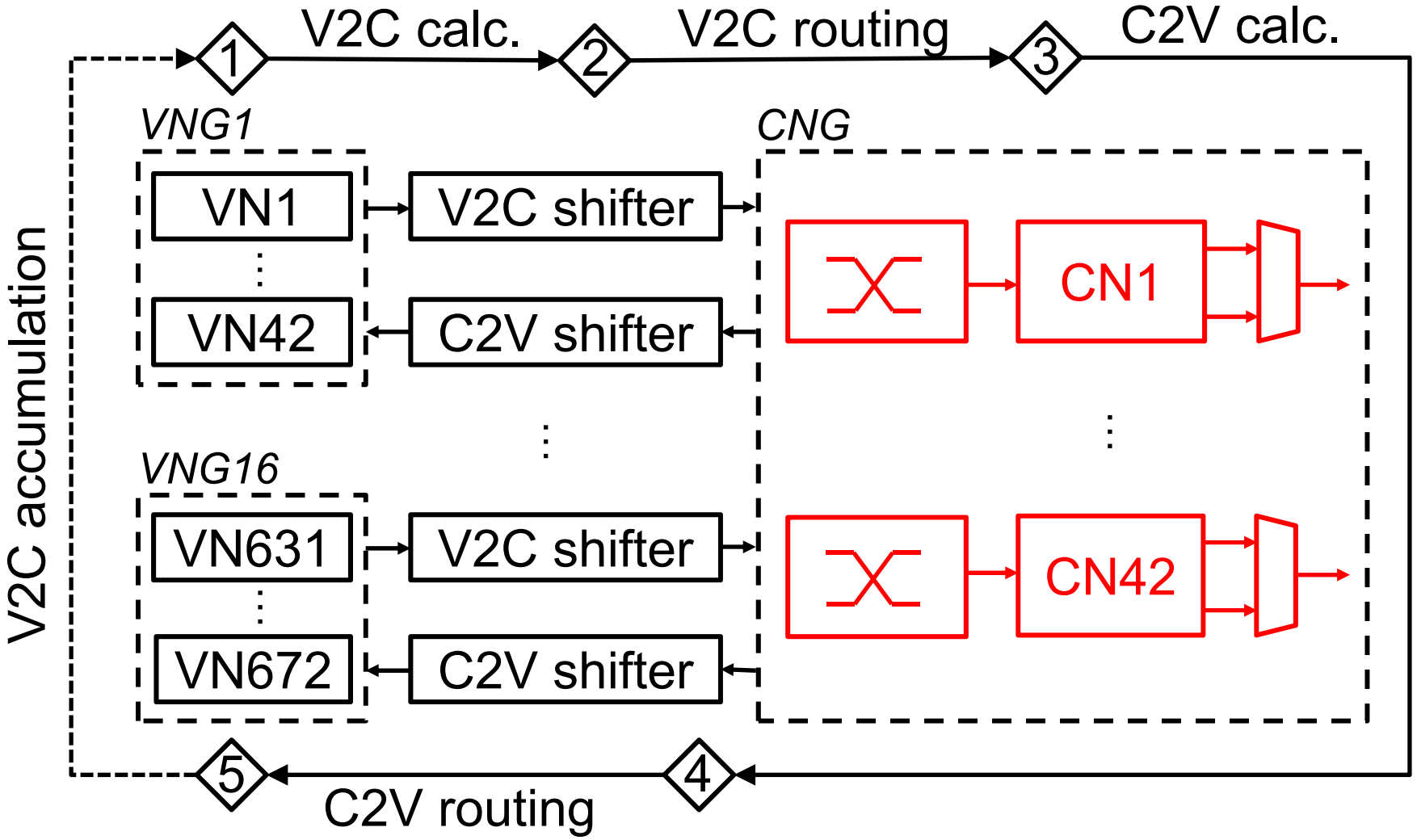

Focus on reducing memory and deep voltage/freq. scaling (VFS)

# Decoder pipelining

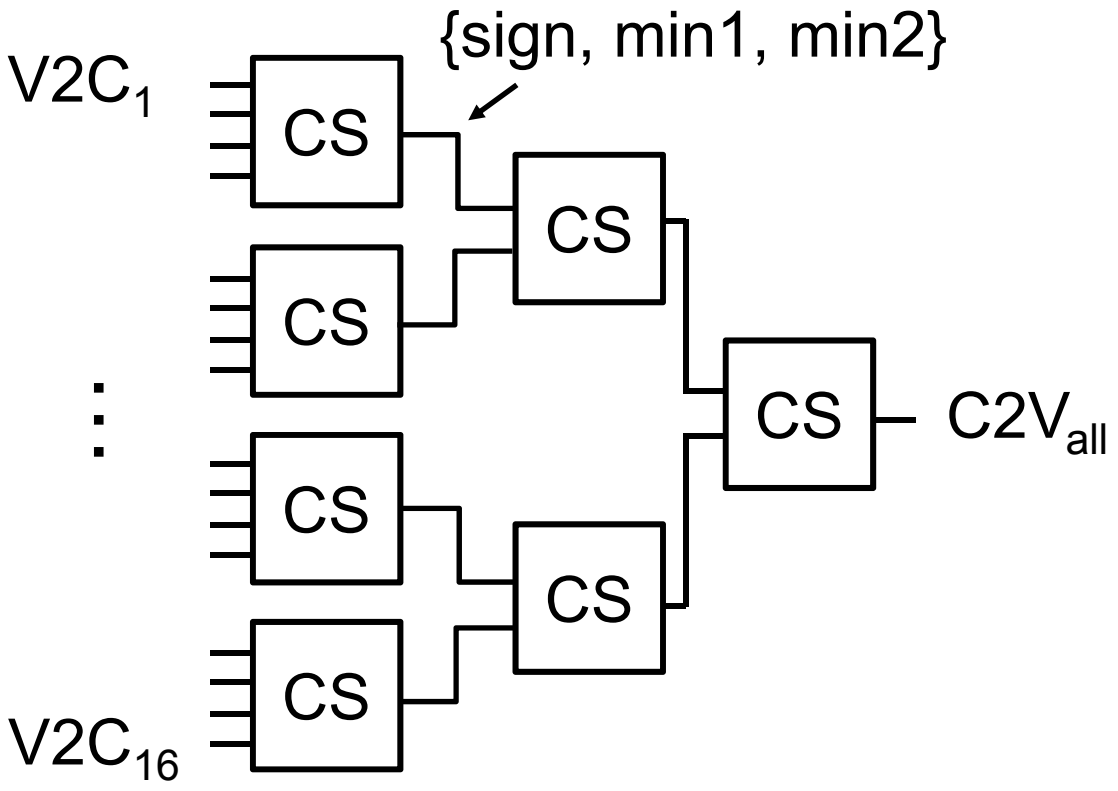



Efficient, deep pipeline reduces critical paths and cycles/it.

# Check node architecture



# Processing a single layer




Tree-based CN has shortest delay

# Exploiting the matrix structure

672 bits

Rate 3/4

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 35 | 19 | 41 | 22 | 40 | 41 | 39 | 6 | 28 | 18 | 17 | 3 | 28 | | | |
| 29 | 30 | 0 | 8 | 33 | 22 | 17 | 4 | 27 | 28 | 20 | 27 | 24 | 23 | | |
| 37 | 31 | 18 | 23 | 11 | 21 | 6 | 20 | 32 | 9 | 12 | 29 | | 0 | 13 | |
| 25 | 22 | 4 | 34 | 31 | 3 | 14 | 15 | 4 | | 14 | 18 | 13 | 13 | 22 | 24 |

Rate 5/8

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20 | 36 | 34 | 31 | 20 | 7 | 41 | 34 | | 10 | 41 | | | | | |
| 30 | 27 | | 18 | | 12 | 20 | 14 | 2 | 25 | 15 | 6 | | | | |
| 35 | | 41 | | 40 | | 39 | | 28 | | | 3 | 28 | | | |
| 29 | | 0 | | | 22 | | 4 | | 28 | | 27 | 24 | 23 | | |
| | 31 | | 23 | | 21 | | 20 | | 9 | 12 | | | 0 | 13 | |
| | 22 | | 34 | 31 | | 14 | | 4 | | | | | | 22 | 24 |

←Layer

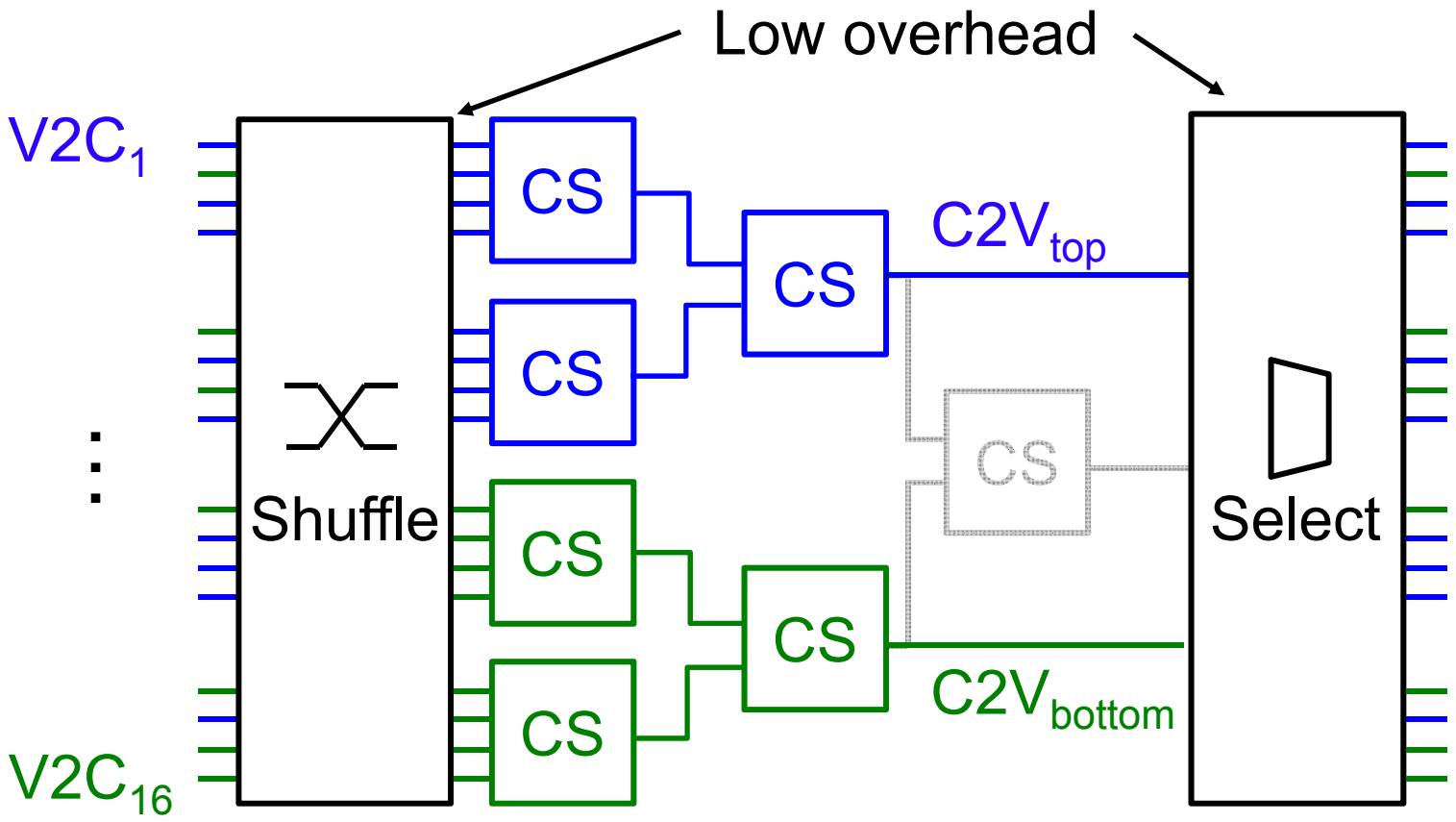
42x42

Rate 1/2

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 40 | | 38 | | 13 | | 5 | | 18 | | | | | | | |
| 34 | | 35 | | 27 | | | 30 | 2 | 1 | | | | | | |
| | 36 | | 31 | | 7 | | 34 | | 10 | 41 | | | | | |
| | 27 | | 18 | | 12 | 20 | | | | 15 | 6 | | | | |
| 35 | | 41 | | 40 | | 39 | | 28 | | | 3 | 28 | | | |
| 29 | | 0 | | | 22 | | 4 | | 28 | | 27 | | 23 | | |
| | 31 | | 23 | | 21 | | 20 | | | 12 | | | 0 | 13 | |
| | 22 | | 34 | 31 | | 14 | | 4 | | | | 13 | | 22 | 24 |

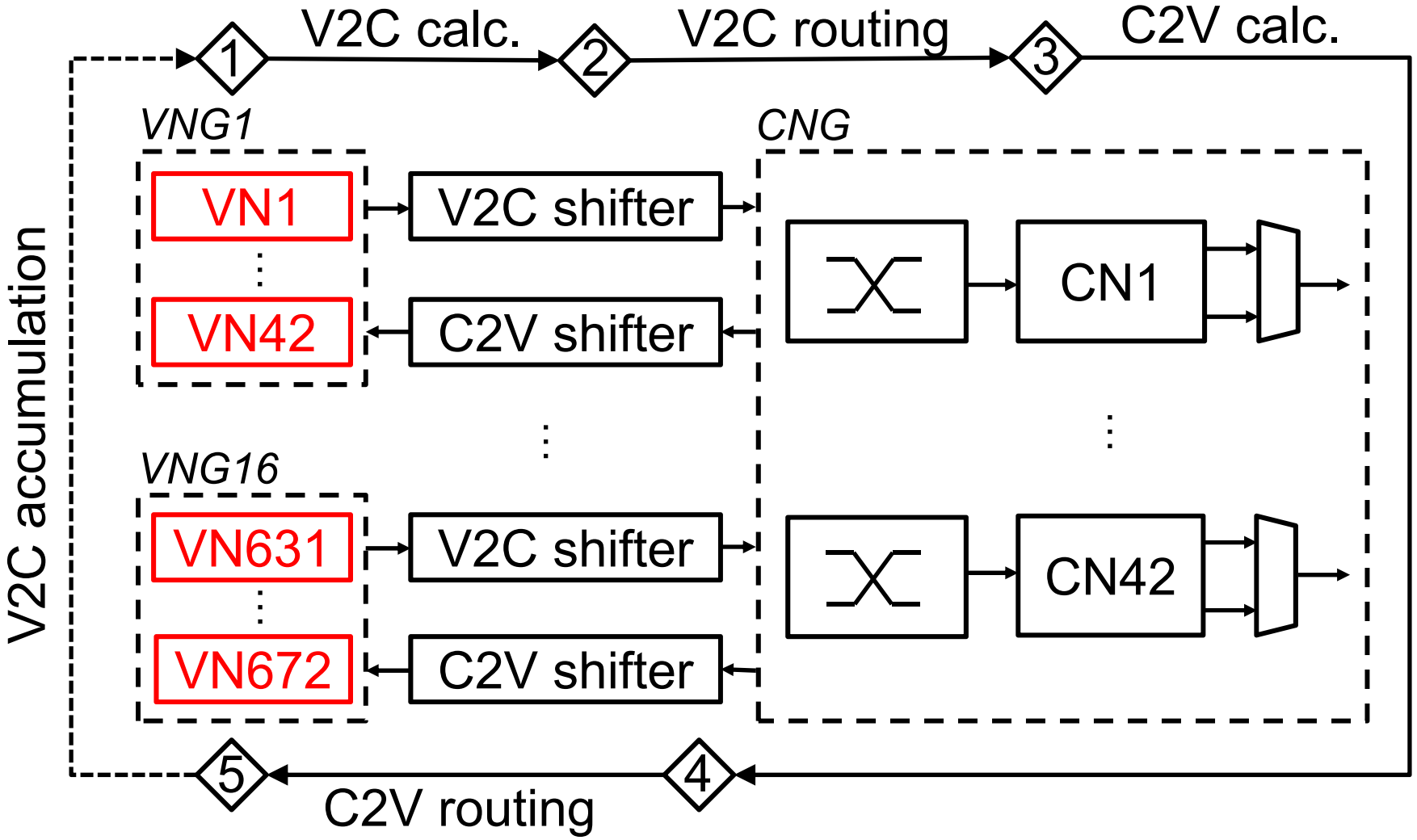
Non-overlapping ½ weight layers can be processed together

# Efficiently reconfigurable CN

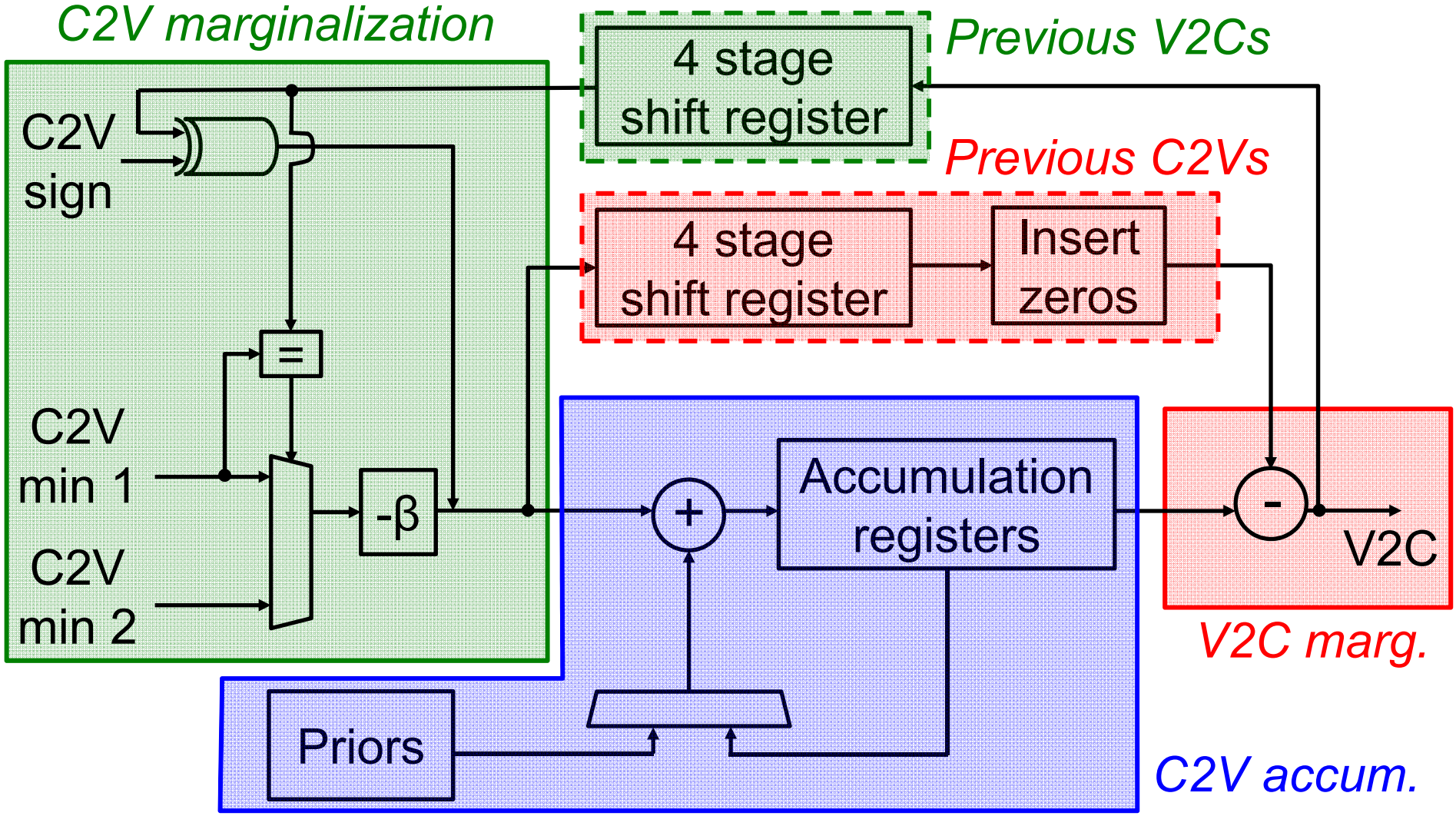



Decreases the cycles/iteration for the worst-case matrix by 50%

# Variable node architecture

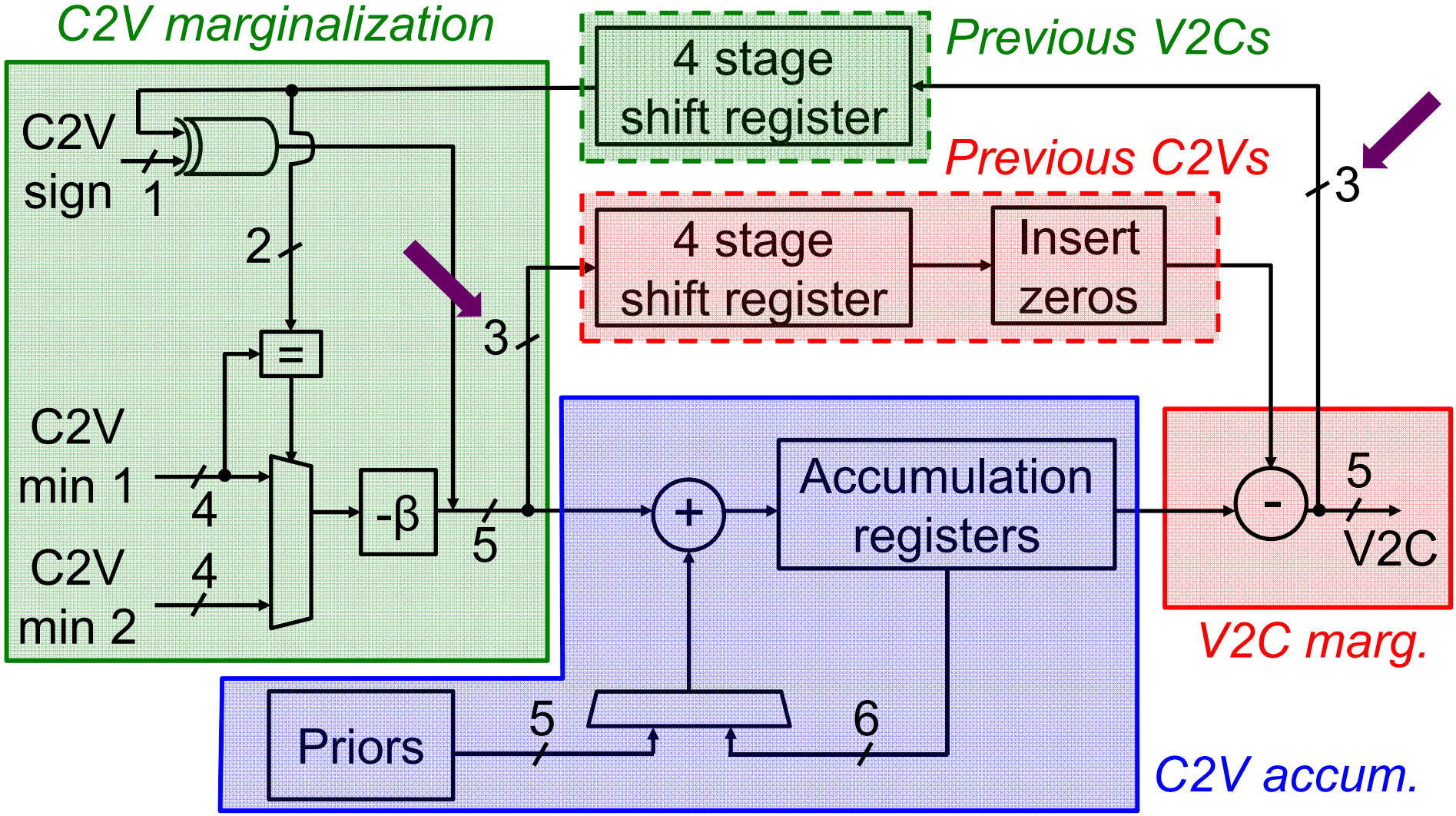


# VN contains majority of memory



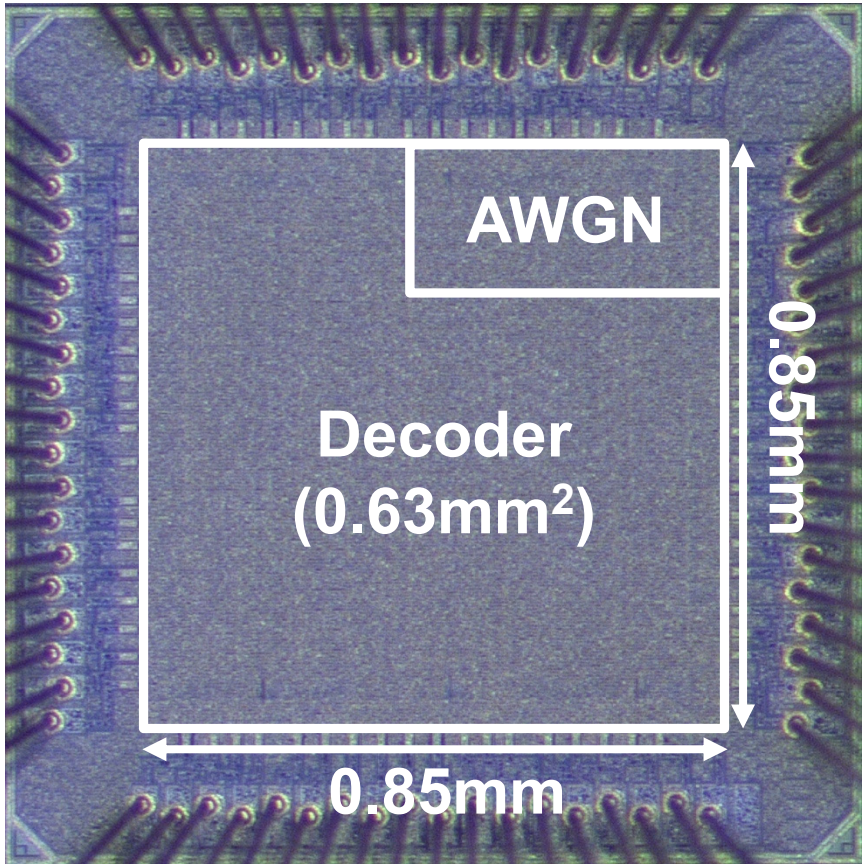

All FF memory: efficient for short blocklengths and low $V_{DD}$

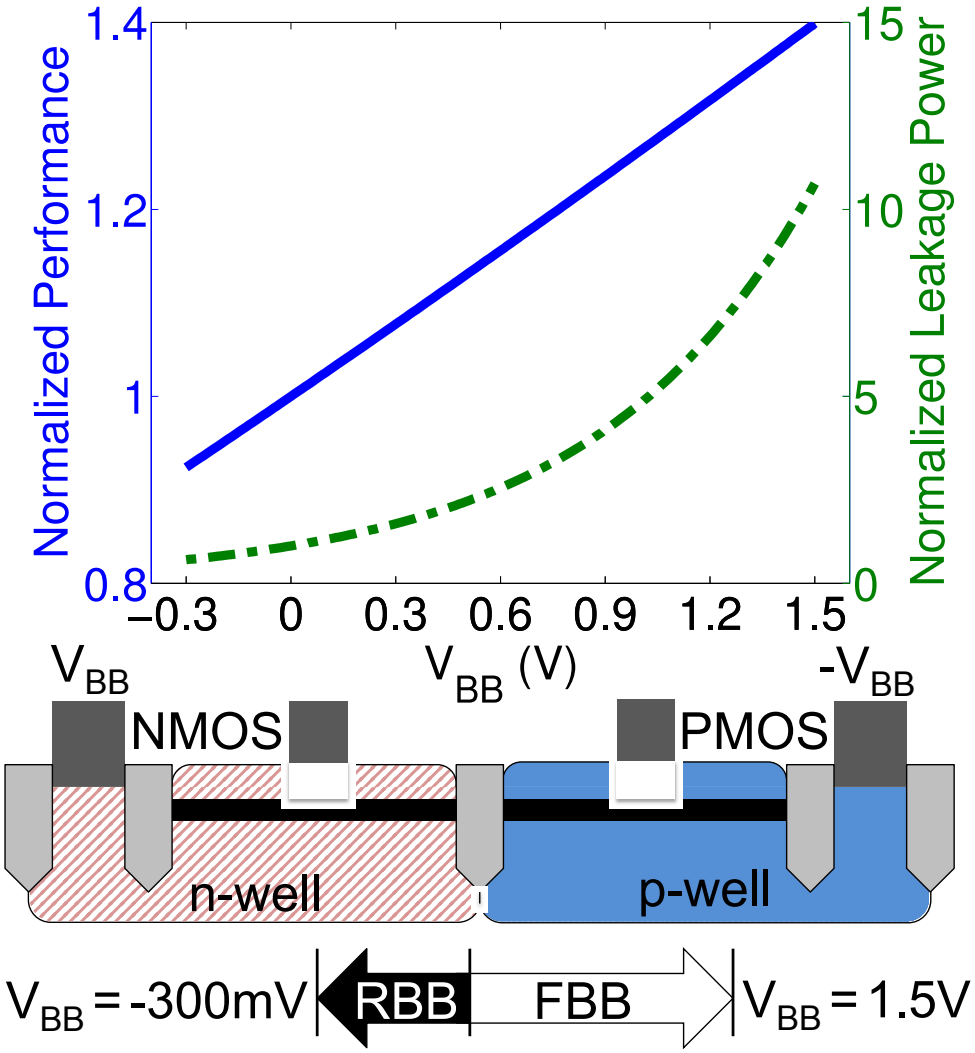
# VN contains majority of memory




Lowers overall power by extra 15%, negligible effect on BER

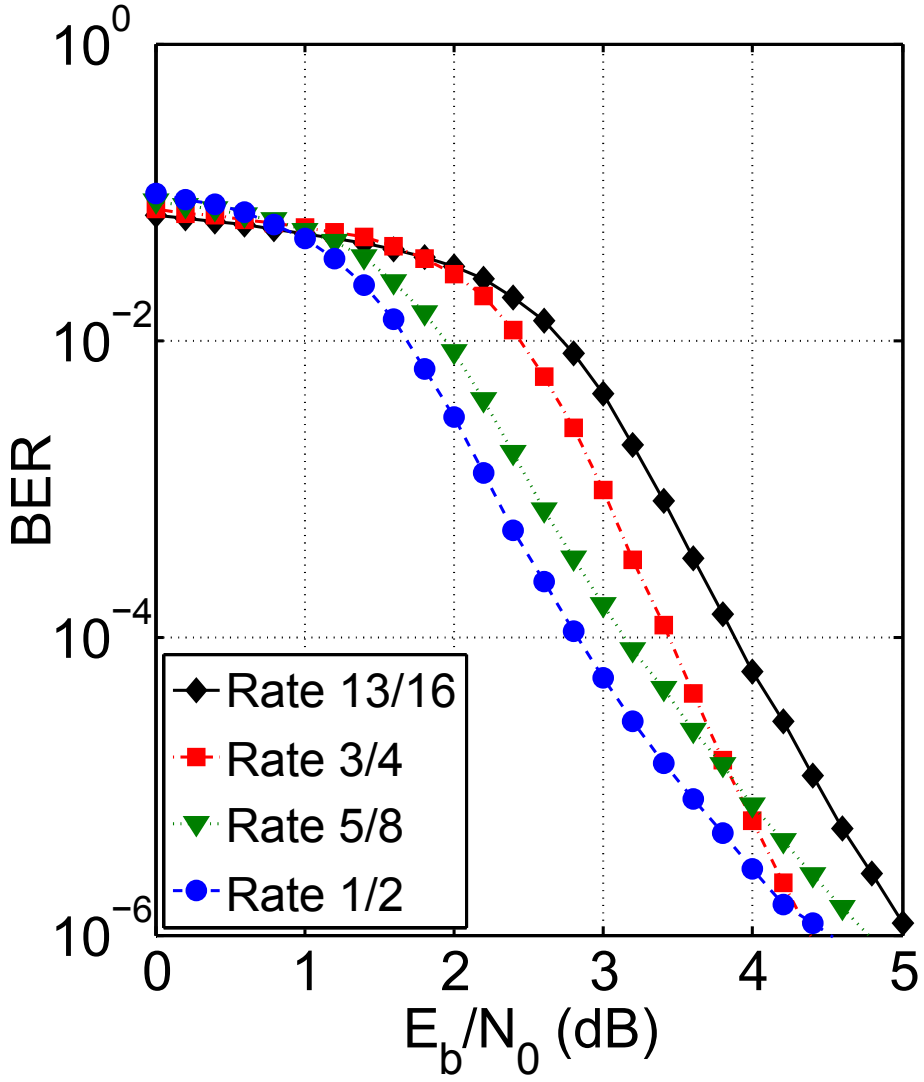
# Die micrograph and FDSOI details

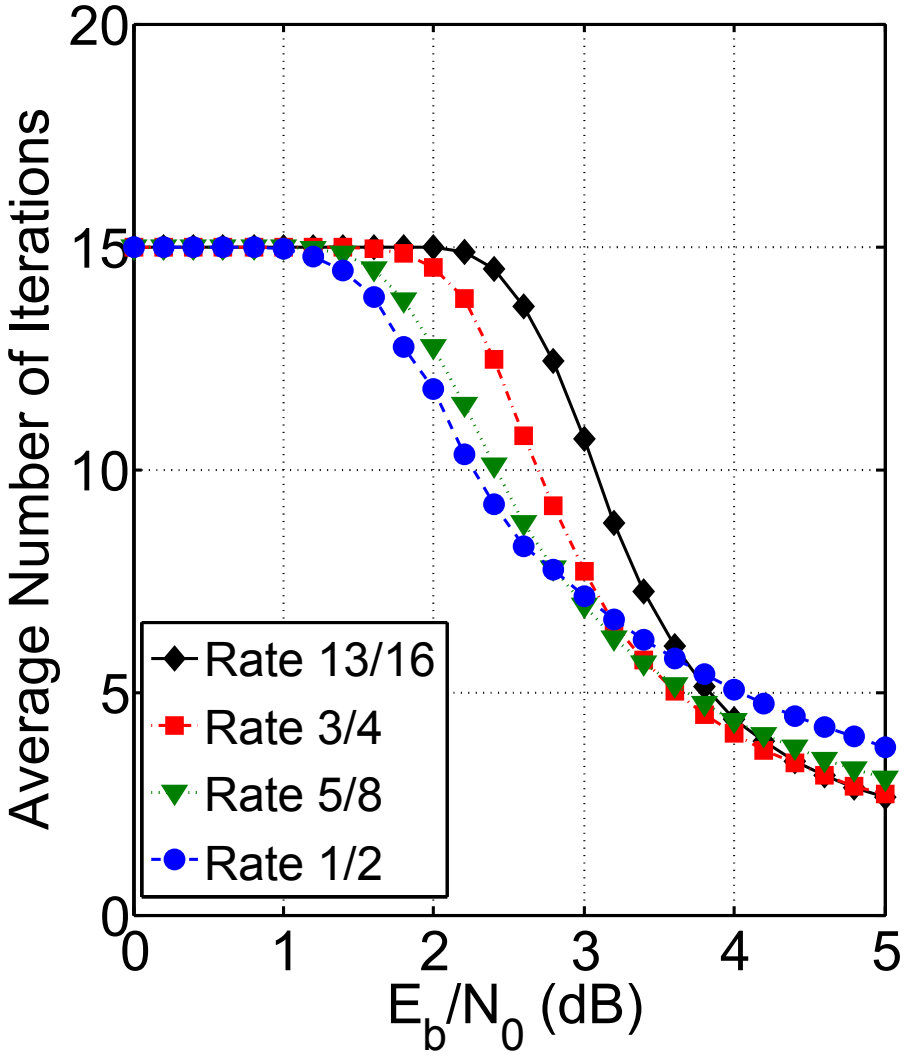



FDSOI body biasing enables additional voltage scaling

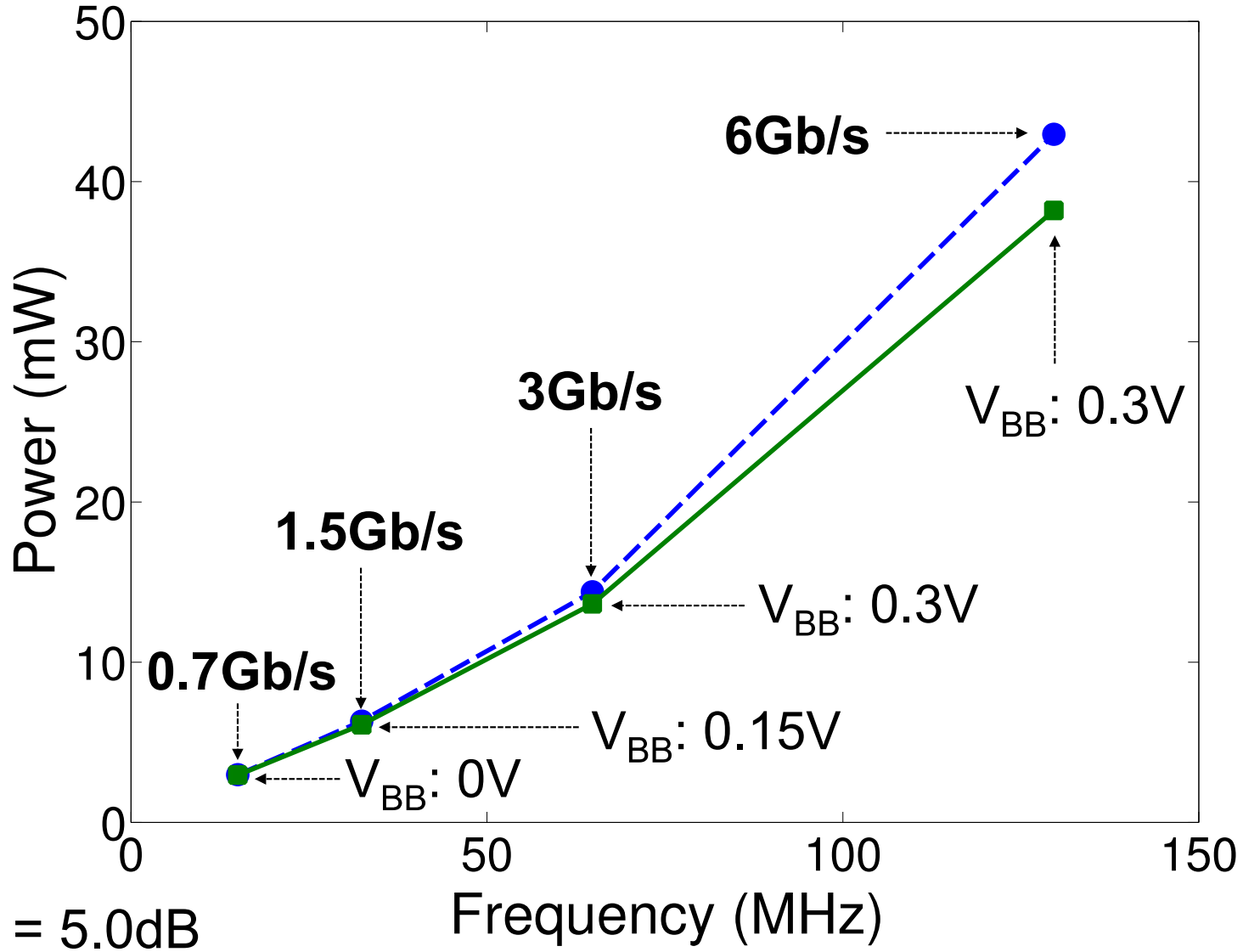
# Measured BER and average iterations*




*15 iterations maximum

# Measured power vs. throughput*




*$E_b/N_0$ = 5.0dB

# Comparison to prior work

| | **This work** | | | VLSI'12 | | | JSSC'12 | | ASSC'11 |
|---|---|---|---|---|---|---|---|---|---|
| Technology | **28nm FDSOI** | | | 65nm bulk | | | 65nm bulk | | 65nm bulk |
| Standard | **802.11ad** | | | 802.11ad | | | 802.15.3c | | 802.16e |
| Blocklength | **672** | | | 672 | | | 672 | | 576-2304 |
| Code rates | **1/2, 5/8, 3/4, 13/16** | | | 1/2 | | | 1/2, 5/8, 3/4, 7/8 | | 1/2, 2/3, 3/4, 5/6 |
| Decoding schedule | **Flooding** | | | Flooding | | | Layered | | Layered |
| Core area ($mm^2$) | **0.63** | | | 1.6 | | | 1.56 | | 3.36 |
| Throughput (Gb/s) | **1.5** | **3** | **6** | 1.5 | 3 | 6 | 3.3 | 5.79 | 1.056 |
| Core supply (V) | **0.51** | **0.58** | **0.71** | 0.51 | 0.64 | 0.94 | 1.00 | 1.00 | 1.20 |
| Memory supply (V) | **-** | **-** | **-** | 0.8 | 0.92 | 1.11 | - | - | - |
| Back bias supply (V) | **0.15** | **0.3** | **0.3** | - | - | - | - | - | - |
| Power (mW) | **6.2** | **13.6** | **38.1** | 37.7 | 106 | 374 | 320 | 400 | 115 |
| Energy efficiency (pJ/decoded bit/it) | **1.1** | **1.3** | **1.8** | 2.5 | 3.5 | 6.2 | N/A | 12.5 | 10.9 |

# Conclusions

- Fully compatible IEEE 802.11ad LDPC decoder implemented in 28nm FDSOI
- Pipelined flooding architecture more efficient than existing layered designs
- FF memory reduces design complexity
- Reduced marginalization precision in VN decreases power by 15%

# Acknowledgments


- BWRC faculty, staff, and sponsors
- STMicroelectronics for chip fabrication
- Brian Zimmer and Jaehwa Kwak
- Andreia Cathelin, Alessandro Cevrero, Vincent Heinrich, Yusuf Leblebici, Nicholas Preyss, Pascal Urard, Engling Yeo, and Zhengya Zhang


# A Static Contention-Free Single-Phase-Clocked 24T Flip-Flop in 45nm for Low-Power Applications

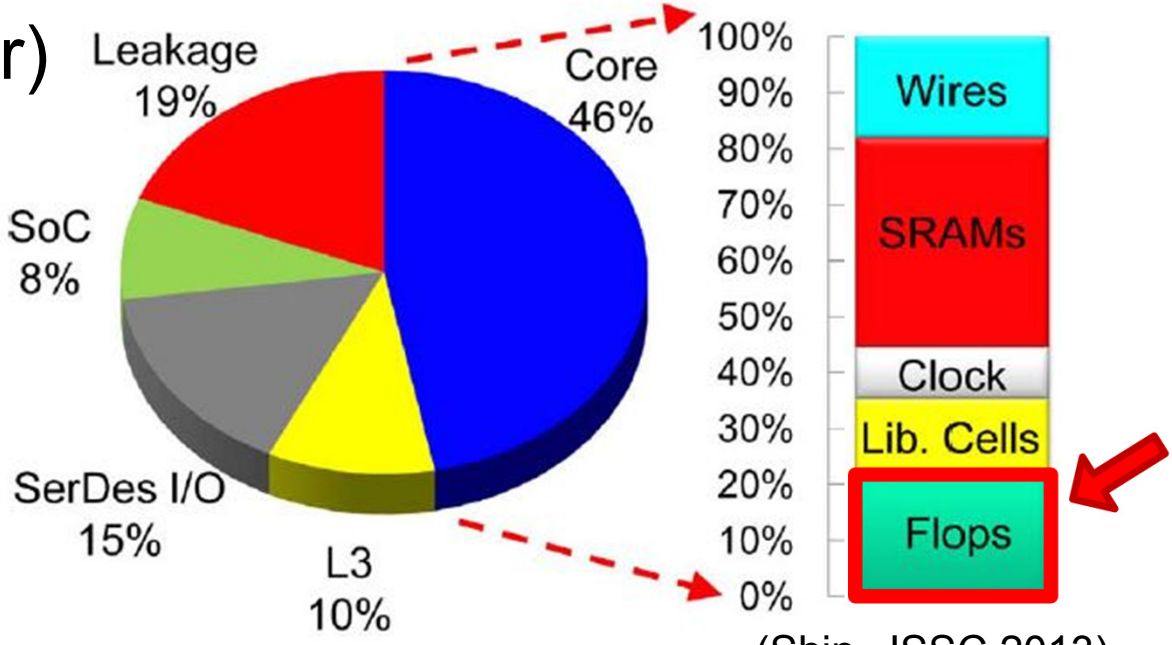
**Yejoong Kim**,
Wanyeong Jung, Inhee Lee, Qing Dong,
Michael Henry, Dennis Sylvester, David Blaauw

yejoong@umich.edu

University of Michigan, Ann Arbor, MI, USA

# Flip-Flops

- **2 Million FFs** in **POWER7™** (Wendel, JSSC 2011)
- **2.5 Million FFs** in **SPARC T4** (Shin, JSSC 2013) (**~20%** of core power)




(Shin, JSSC 2013)

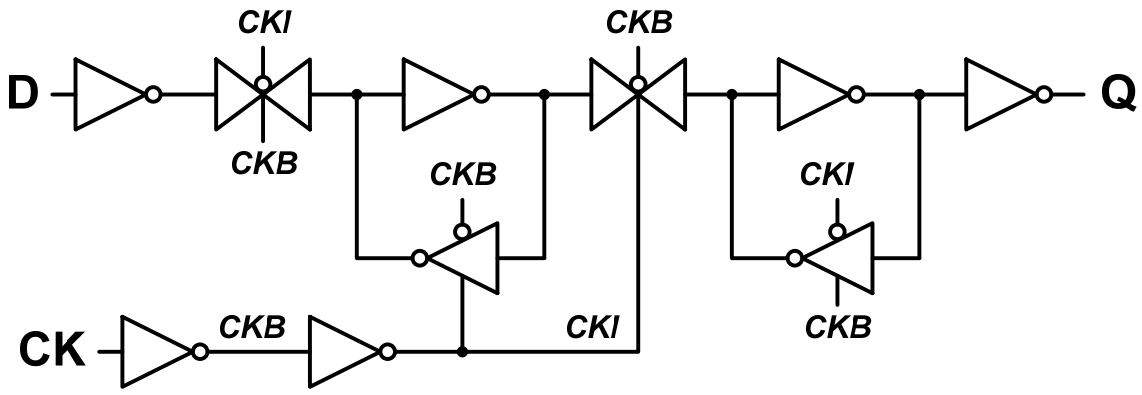
- **Flip-Flops (FFs)** significantly affect **Area**, **Power**, and **Performance** (Warnock, ISSCC 2010)
- **A Major Bottleneck** in **$V_{DD}$ scaling** (Kaul, DAC 2012)

# Requirements for a Good Flip-Flop

- For Maximum Robustness:
  - **Static** (No data is stored on a dynamic node)
  - **Contention-Free** (No fight between NMOS & PMOS)
- For Low Power Consumption:
  - **Single-Phase** Clock Operation
    (No need for ***CKB***; no internal clock inverter)
- In Addition:
  - Fast Speed
  - **Compact or Small Area**

# Conventional Flip-Flops: TGFF

| | TGFF | | | |
|---|---|---|---|---|
| Static Operation | YES | | | |
| Contention-Free | YES | | | |
| Single-Phase | NO | | | |
| Device Count | 24 | | | |

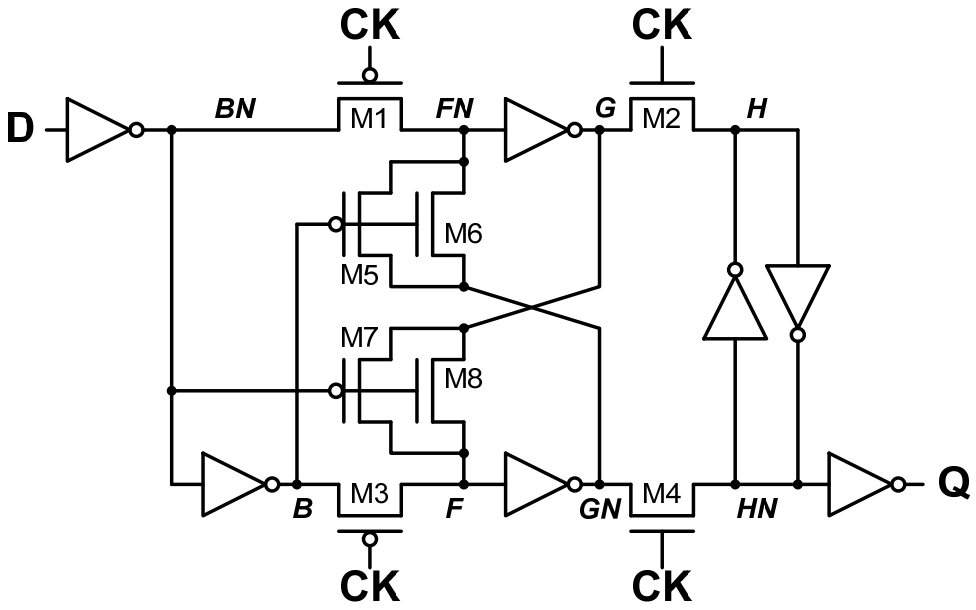



**< TGFF (Transmission-Gate FF) >**

- Found in most Standard Cell Libraries
- Static, Contention-Free
- **No Single-Phase**
- 24 Devices

# Conventional Flip-Flops: ACFF

| | TGFF | ACFF [3] | | |
|---|---|---|---|---|
| **Static Operation** | YES | YES | | |
| **Contention-Free** | YES | **NO** | | |
| **Single-Phase** | **NO** | YES | | |
| **Device Count** | 24 | 22 | | |

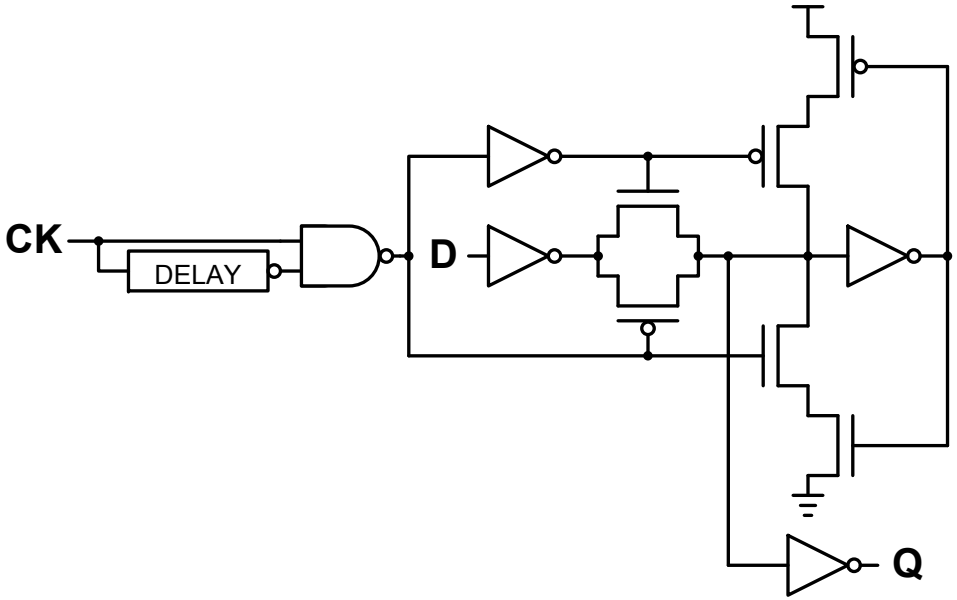



< ACFF (Adaptive-Coupling FF) >

- Teh, ISSCC 2011
- Static, Single-Phase
- **Contention exists in slave latch**
- 22 Devices

# Conventional Flip-Flops: TGPL

| | TGFF | ACFF [3] | TGPL [4] |
|---|---|---|---|
| Static Operation | YES | YES | YES |
| Contention-Free | YES | NO | YES |
| Single-Phase | NO | YES | NO |
| Device Count | 24 | 22 | 28 |

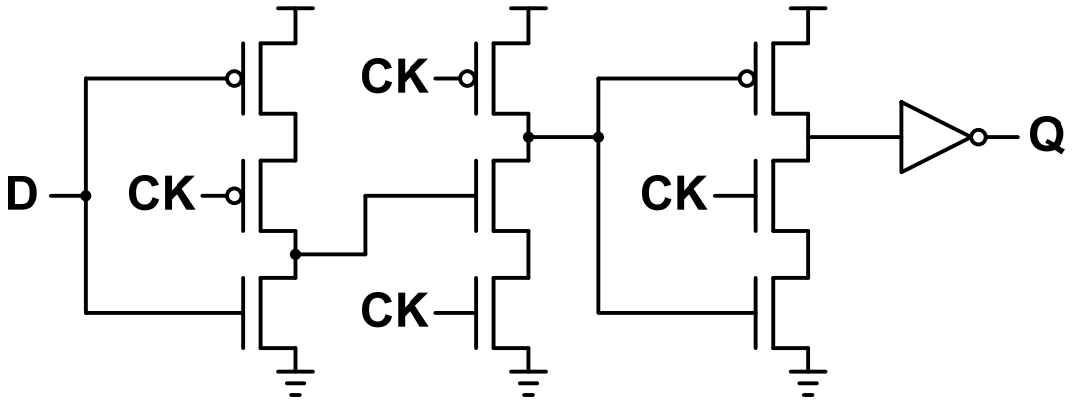



- Naffziger, JSSC 2002
- **Pulsed Operation**
- Contention-Free
- 28 Devices

**< TGPL (TR-Gate Pulsed-Latch) >**

# Conventional Flip-Flops: TSPC

| | TGFF | ACFF [3] | TGPL [4] | TSPC [5] |
|---|---|---|---|---|
| **Static Operation** | YES | YES | YES | **NO** |
| **Contention-Free** | YES | **NO** | YES | YES |
| **Single-Phase** | **NO** | YES | **NO** | YES |
| **Device Count** | 24 | 22 | 28 | 11 |

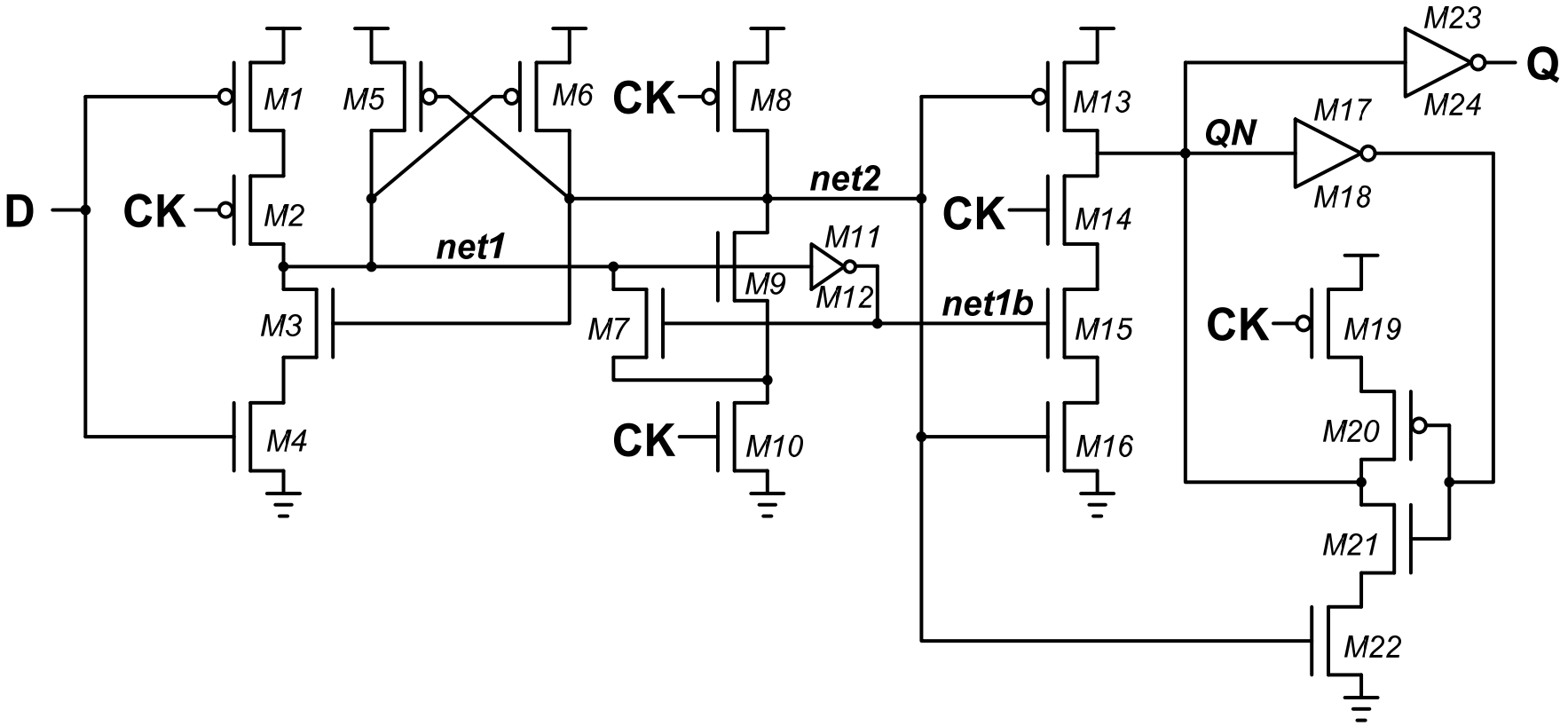



- Yuan, JSSC 1989
- **Dynamic**
- Single-Phase
- Contention-Free
- 11 Devices
- Glitch @ D=0, $Q_{PREV}$=0

**< TSPC (True Single-Phase Clock FF) >**

# Proposed Flip-flop: S²CFF

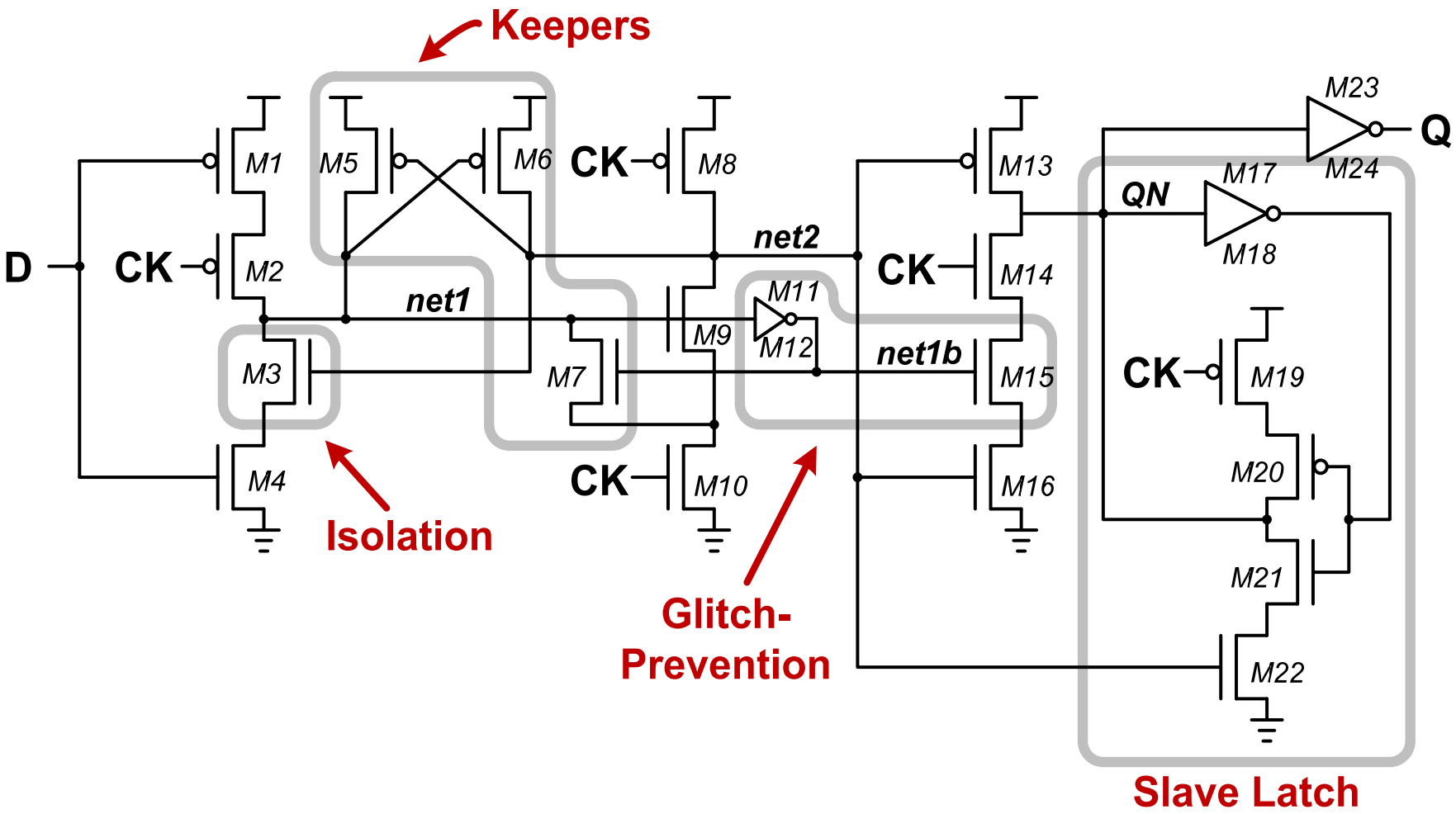
- ***S***tatic ***S***ingle-phase ***C***ontention-free ***F***lip-***F***lop (**S²CFF**)




- Total **24 Devices** (same as in TGFF)

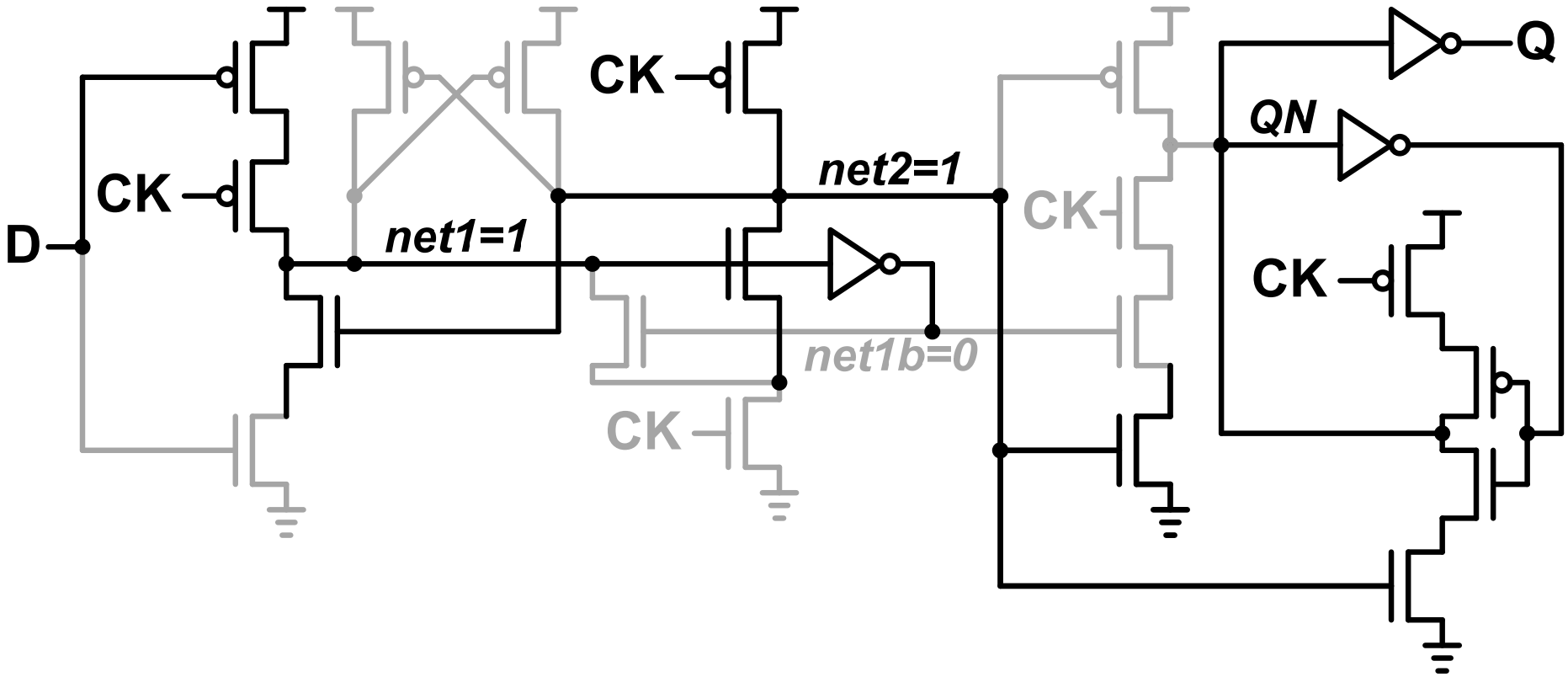
# Proposed Flip-flop: S²CFF

- *S*tatic *S*ingle-phase *C*ontention-free *F*lip-*F*lop (**S²CFF**)



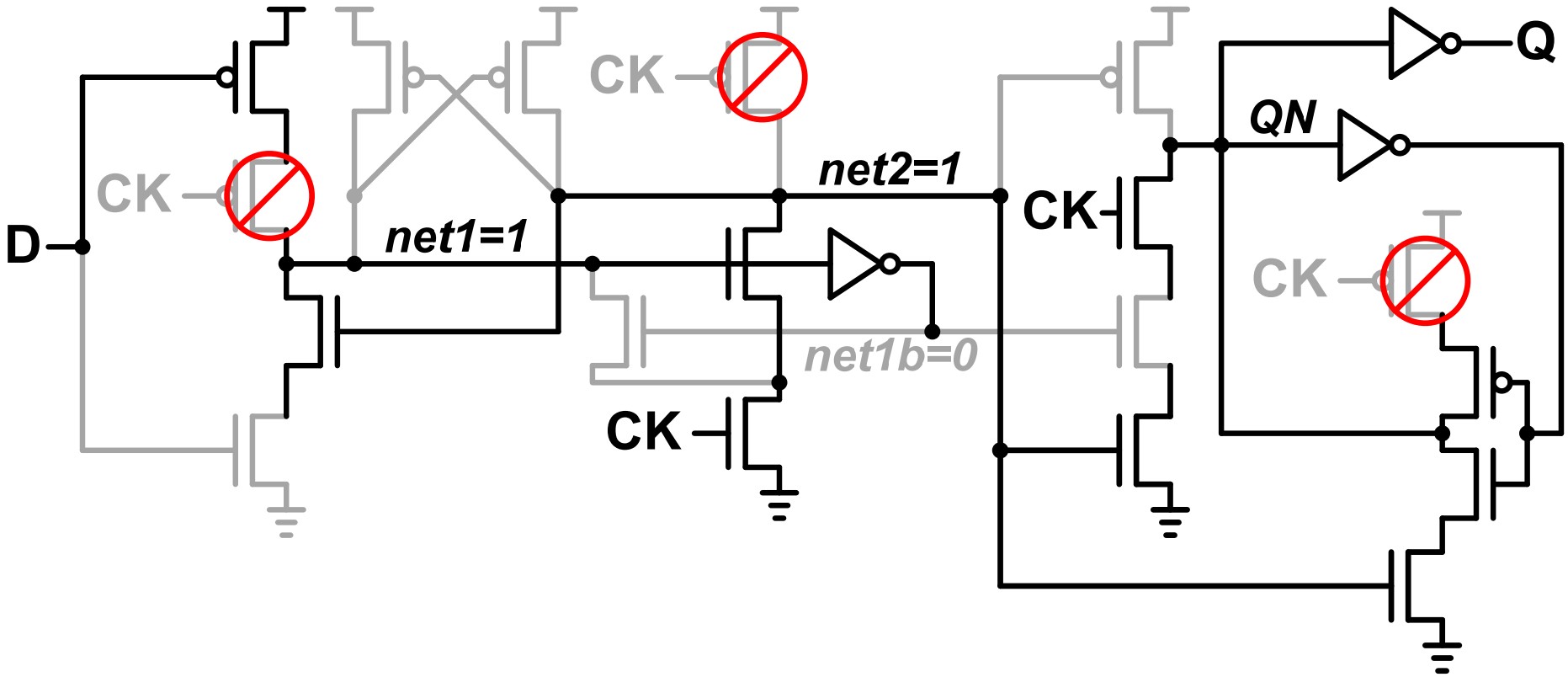

- Total **24 Devices** (same as in TGFF)

# S²CFF Operation: D=0 & CK=0



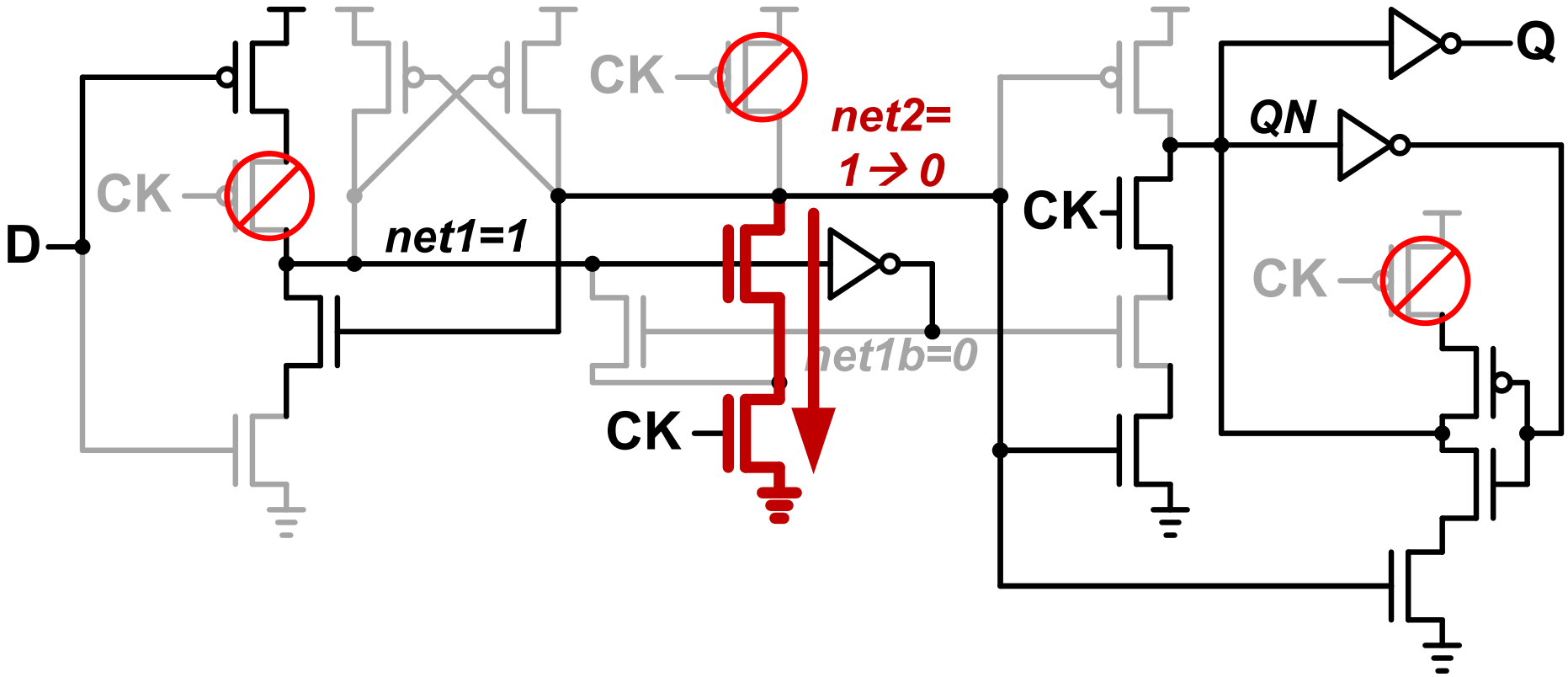

- ***net1*** stores **D-bar** (=1)
- ***net2*** becomes precharged (=1)

# S²CFF Operation: D=0 & CK=0→1 (1/5)



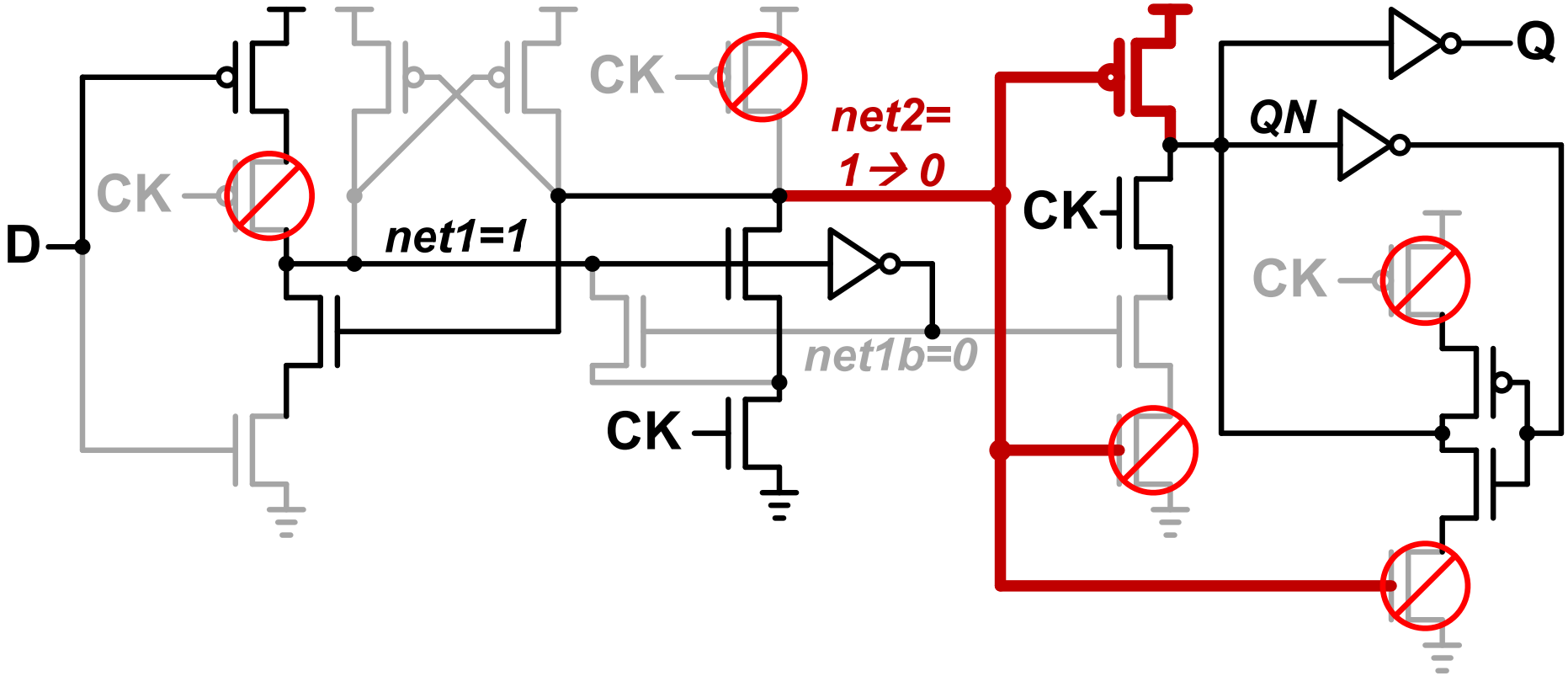

- Clocked devices become **OFF**
- ***net2*** is discharged, updates ***QN*** (and **Q**), isolates **D**
- Feedback between ***net1*** & ***net2***

# S²CFF Operation: D=0 & CK=0→1 (2/5)



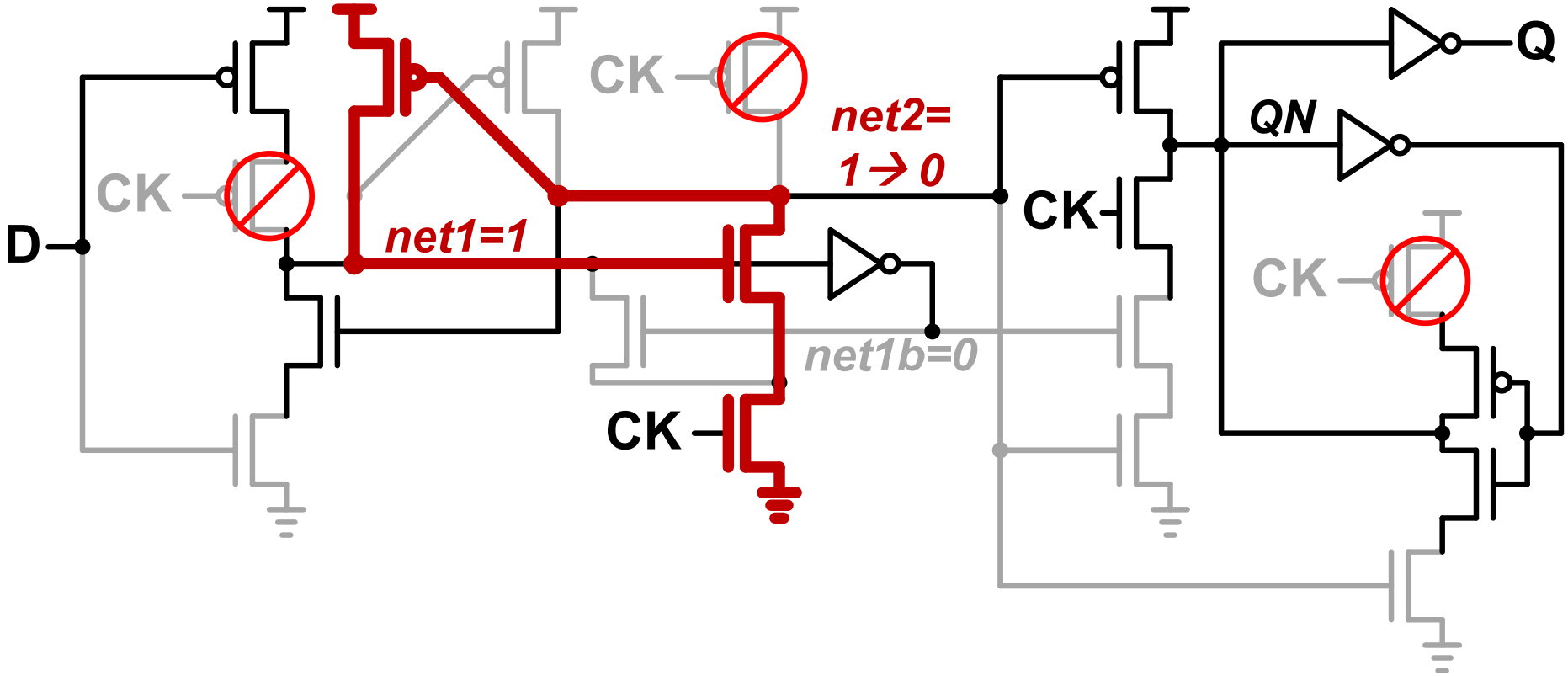

- Clocked devices become **OFF**
- ***net2*** is discharged, updates ***QN*** (and **Q**), isolates **D**
- Feedback between ***net1*** & ***net2***

# S²CFF Operation: D=0 & CK=0→1 (3/5)



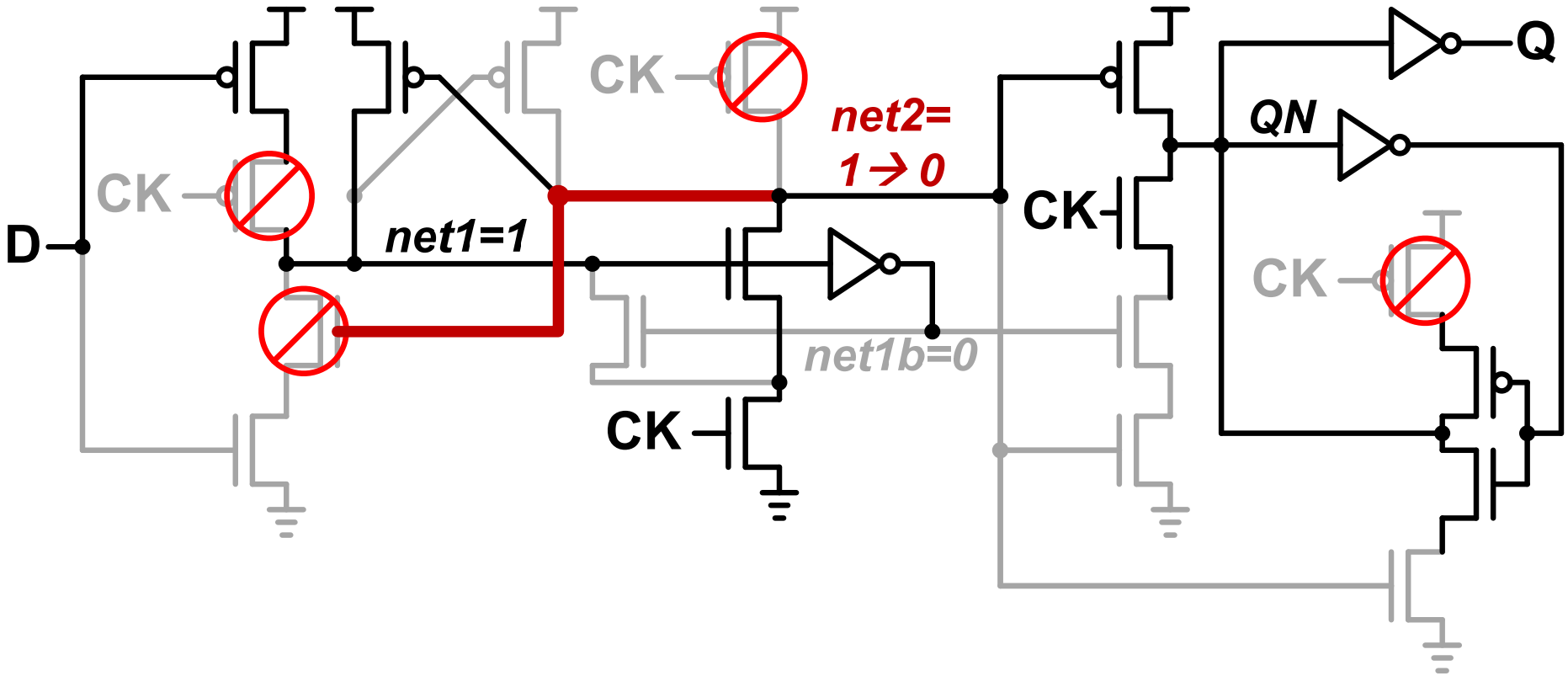

- Clocked devices become **OFF**
- ***net2*** is discharged, updates ***QN*** (and **Q**), isolates **D**
- Feedback between ***net1*** & ***net2***

# S²CFF Operation: D=0 & CK=0→1 (4/5)



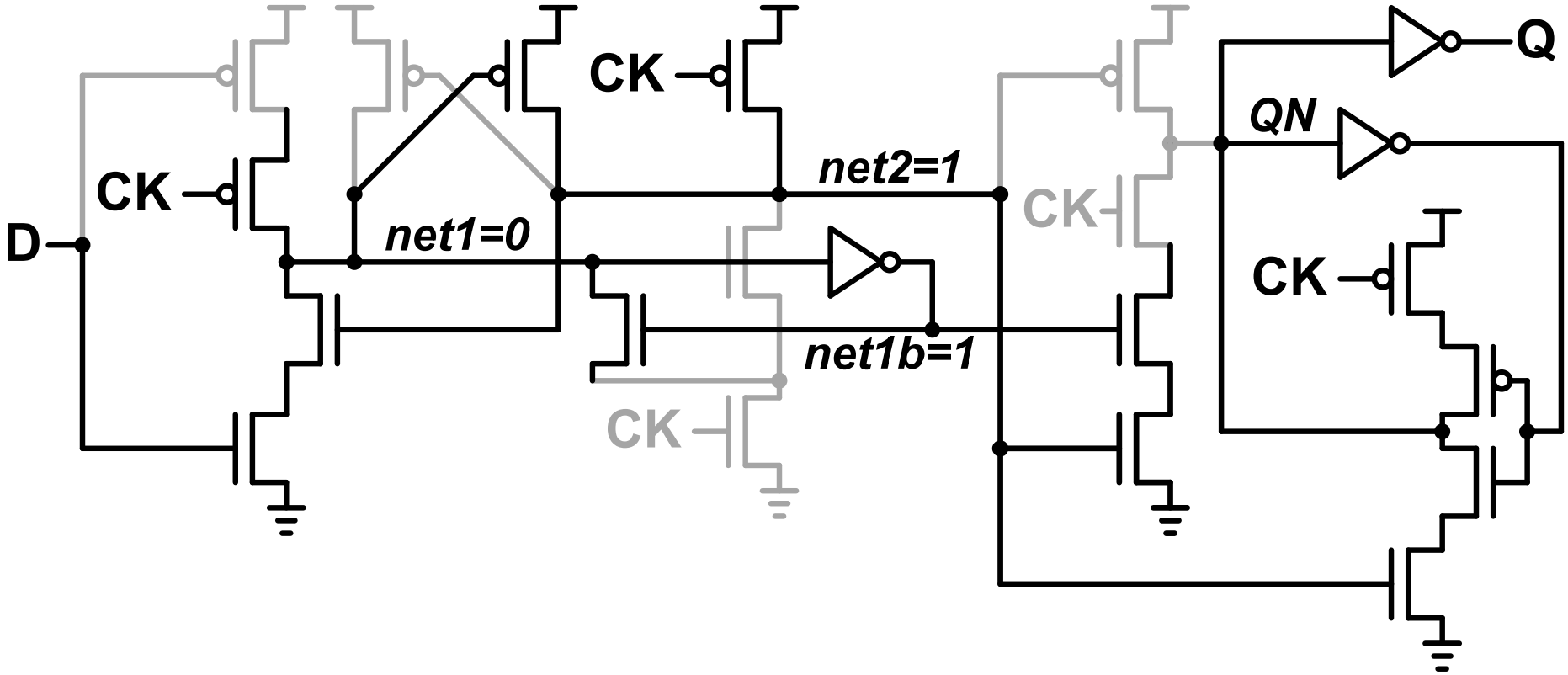

- Clocked devices become **OFF**
- ***net2*** is discharged, updates ***QN*** (and **Q**), isolates **D**
- Feedback between ***net1*** & ***net2***

# S²CFF Operation: D=0 & CK=0→1 (5/5)



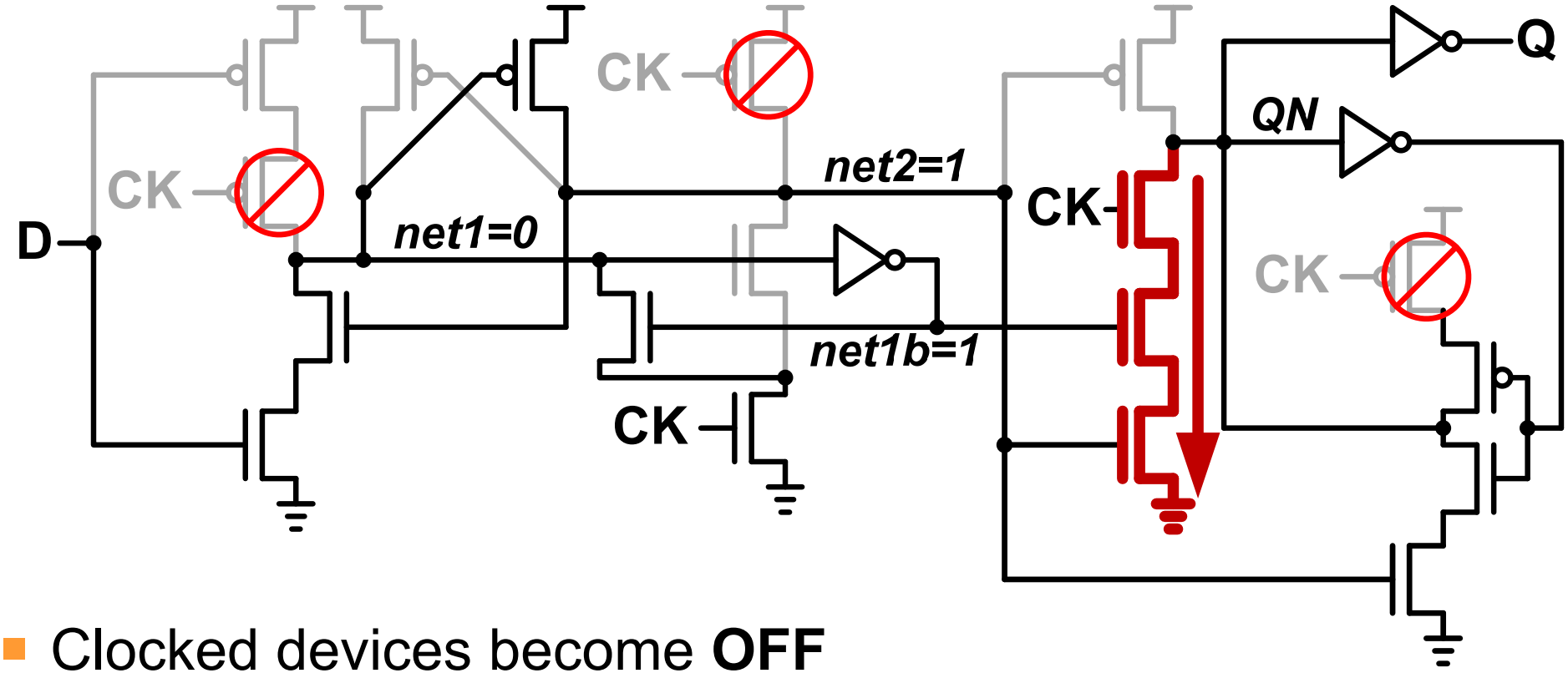

- Clocked devices become **OFF**
- ***net2*** is discharged, updates ***QN*** (and **Q**), isolates **D**
- Feedback between ***net1*** & ***net2***

# S²CFF Operation: D=1 & CK=0



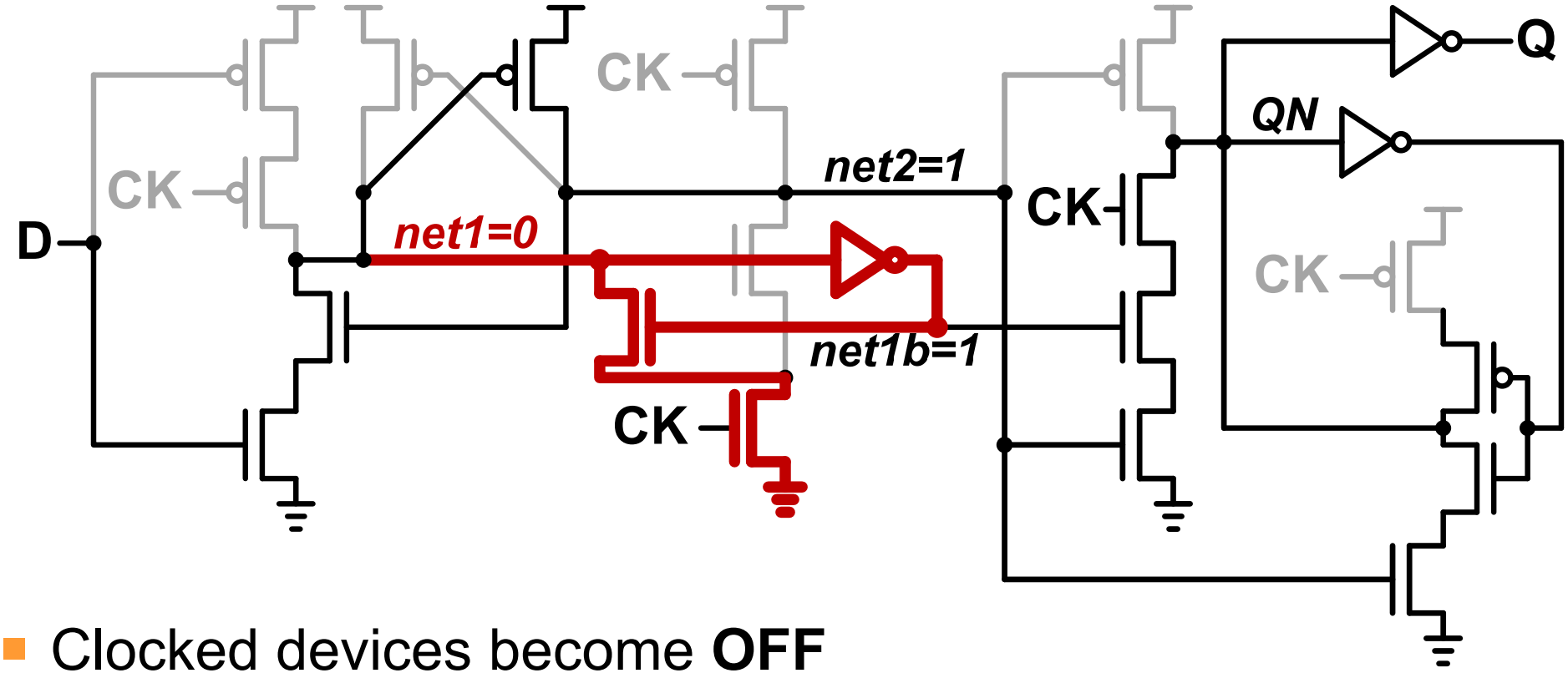

- ***net1*** stores **D-bar** (=0)
- ***net2*** becomes precharged (=1)

# S²CFF Operation: D=1 & CK=0→1 (1/3)



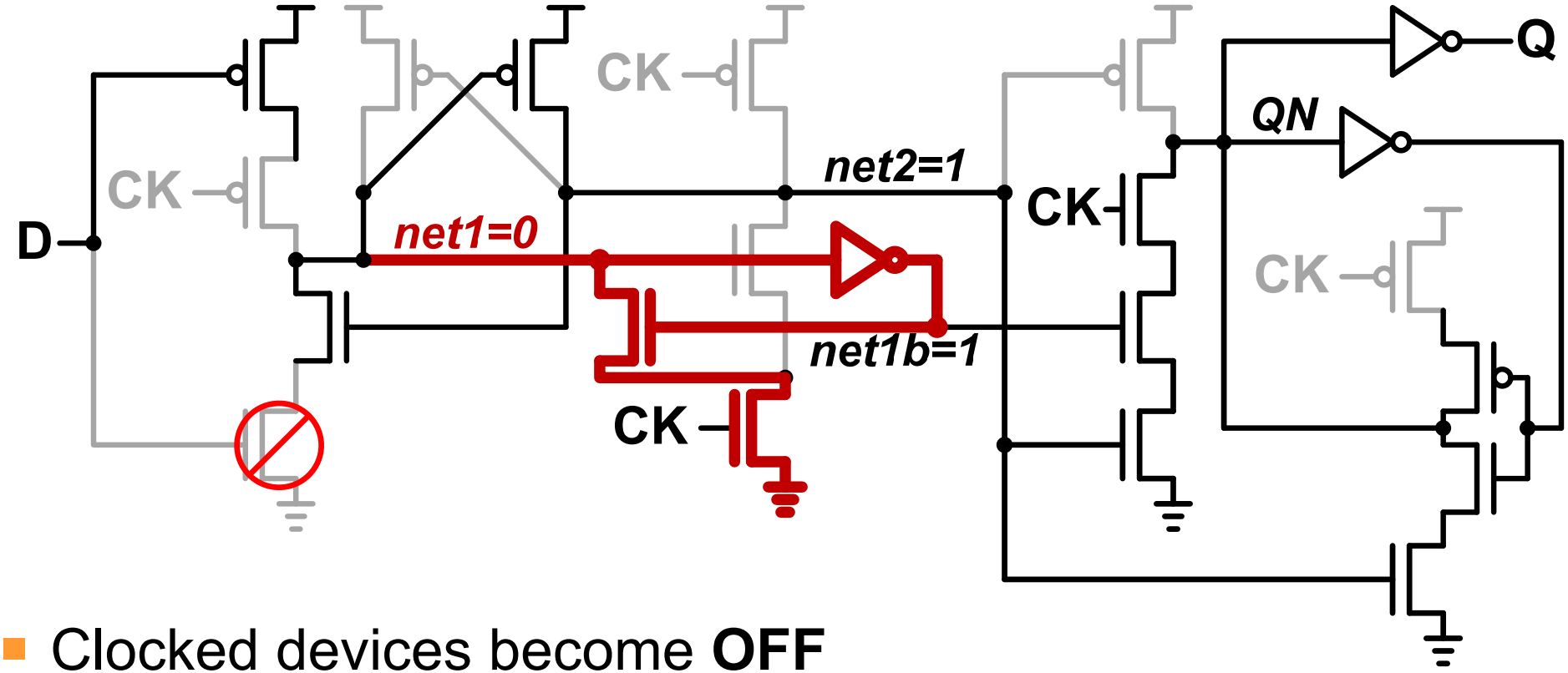

- Clocked devices become **OFF**
- ***net2*** stays high, updates ***QN***
- Feedback keeps ***net1*** low
- Static, single-phase, no contention

# S²CFF Operation: D=1 & CK=0→1 (2/3)



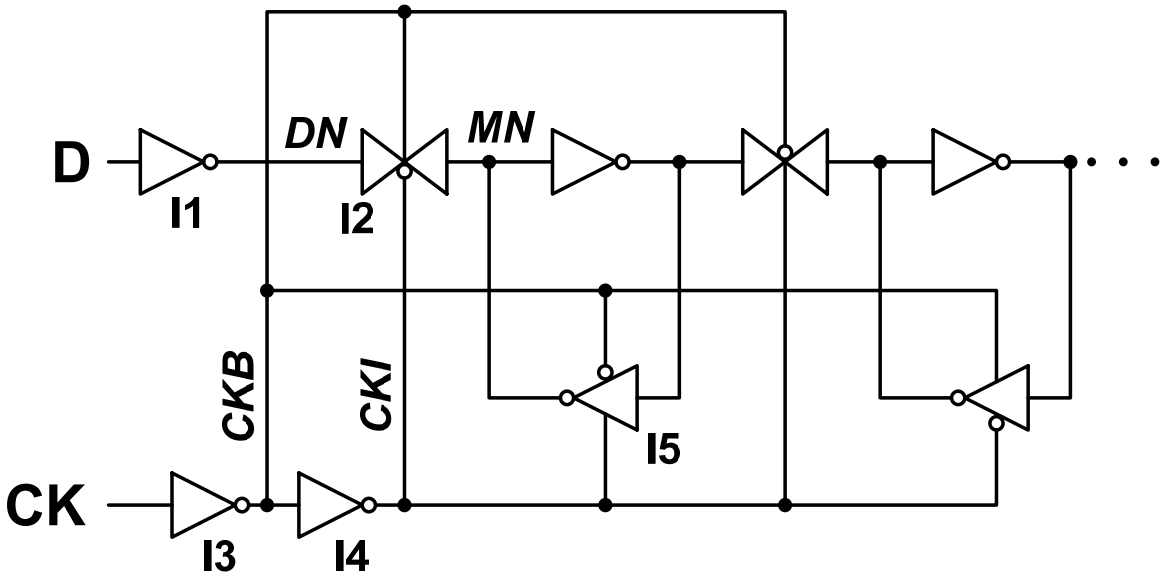

- Clocked devices become **OFF**
- ***net2*** stays high, updates ***QN***
- Feedback keeps ***net1*** low
- Static, single-phase, no contention

# S²CFF Operation: D=1 & CK=0→1 (3/3)



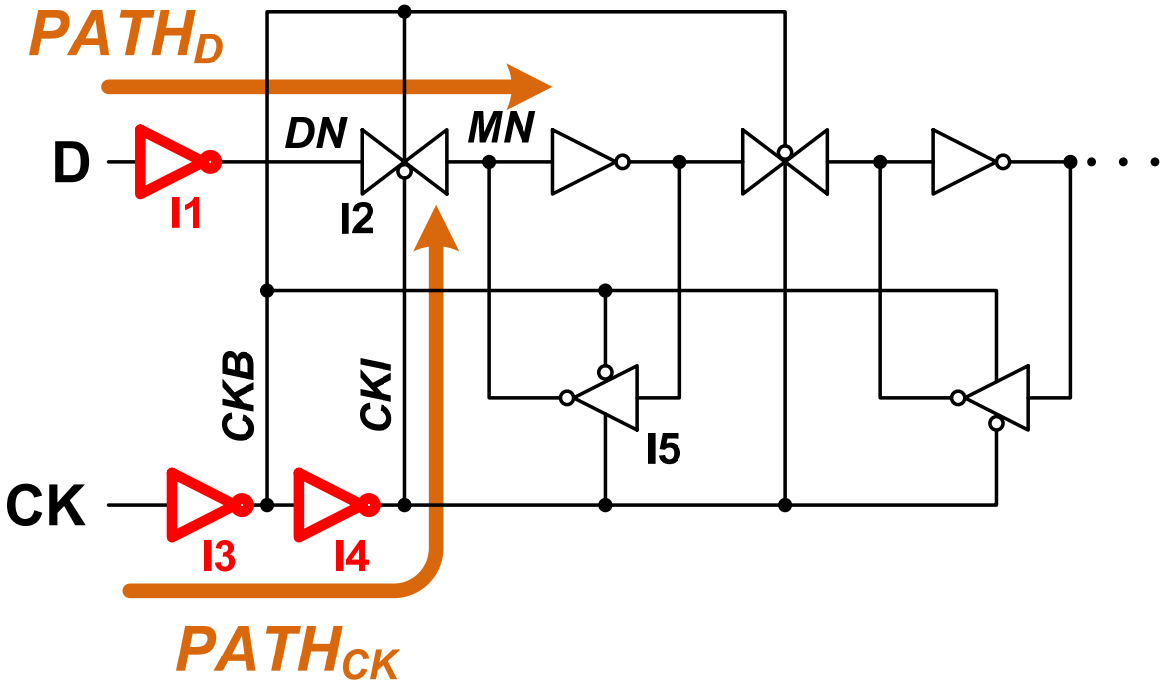

- Clocked devices become **OFF**
- ***net2*** stays high, updates ***QN***
- Feedback keeps ***net1*** low
- Static, single-phase, no contention

# Worst-Case Hold-Time Path in TGFF



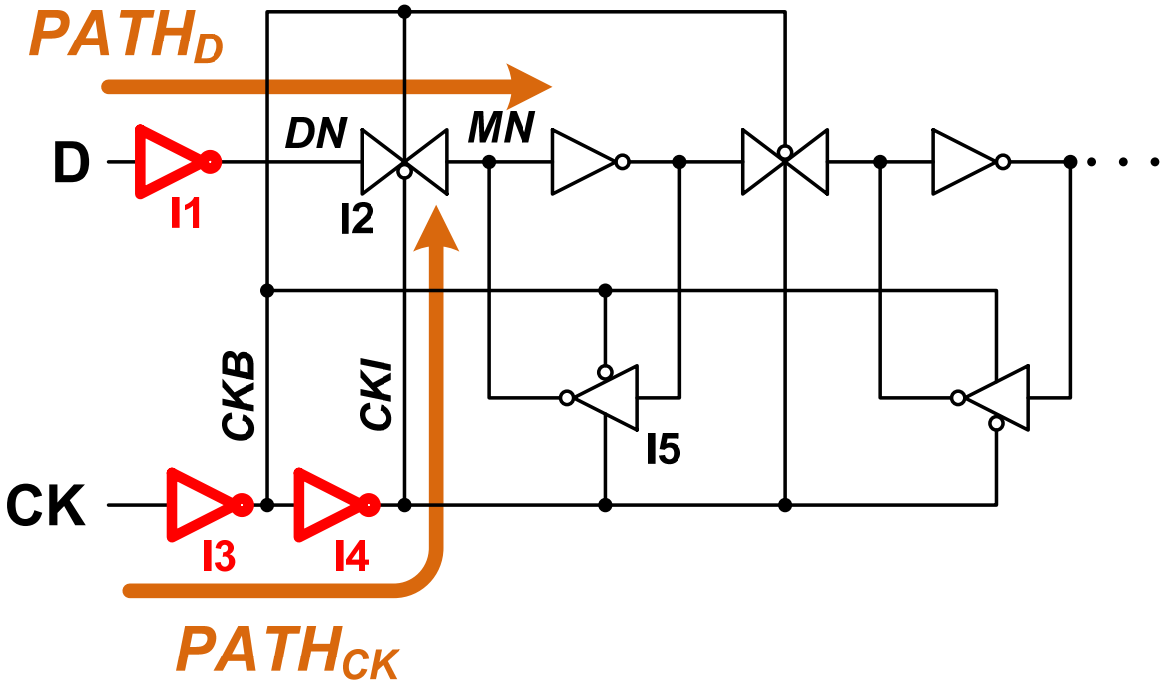

- **Worst-Case $T_{HOLD}$ @ D: 1→0**
- $T_{HOLD}$ dictated by **mismatches** among **I1**, **I3**, **I4**.
- Degraded $T_{HOLD}$ variation at **Low $V_{DD}$**

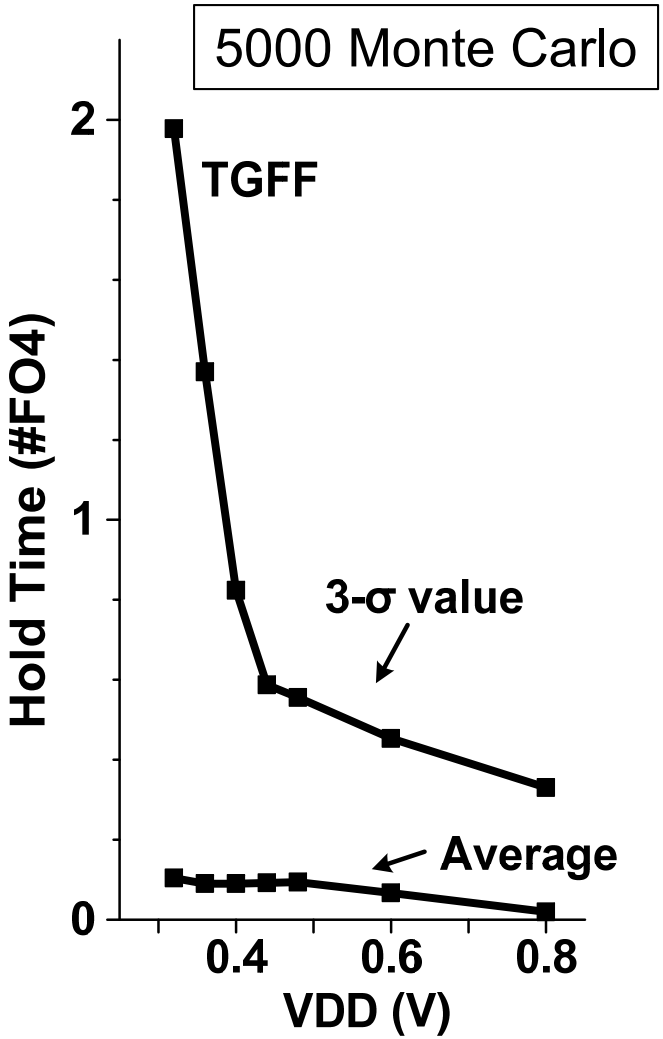
# Worst-Case Hold-Time Path in TGFF




- **Worst-Case $T_{HOLD}$ @ D: 1→0**
- $T_{HOLD}$ dictated by **mismatches** among **I1**, **I3**, **I4**.
- Degraded $T_{HOLD}$ variation at **Low $V_{DD}$**

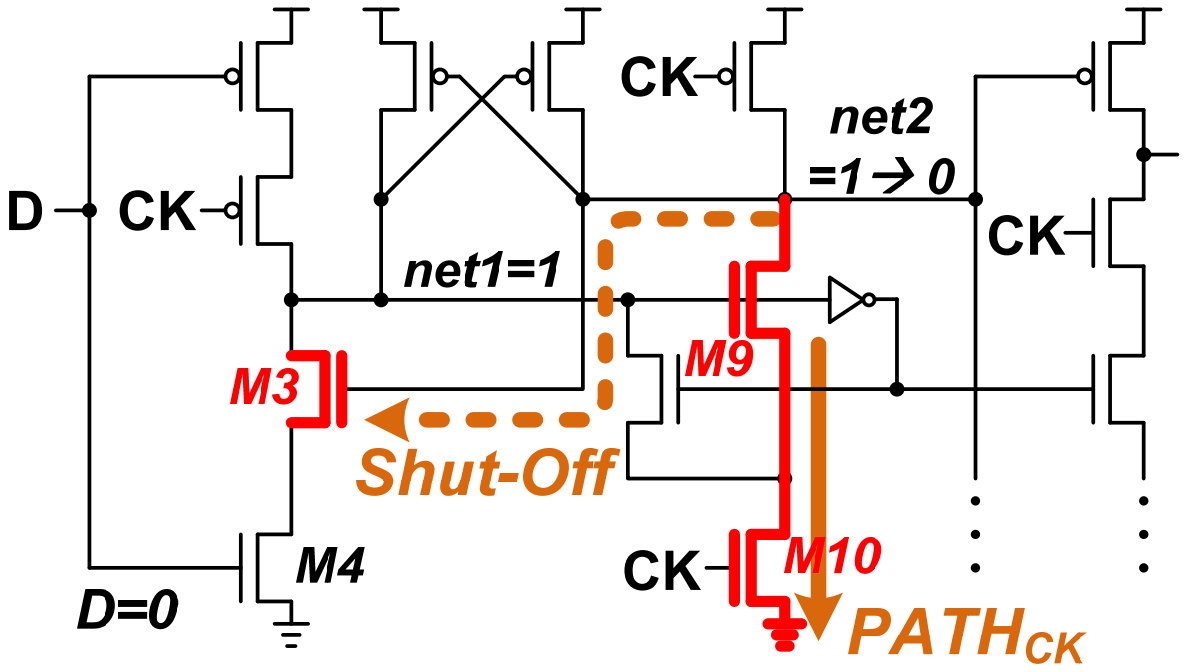
# Worst-Case Hold-Time Path in TGFF

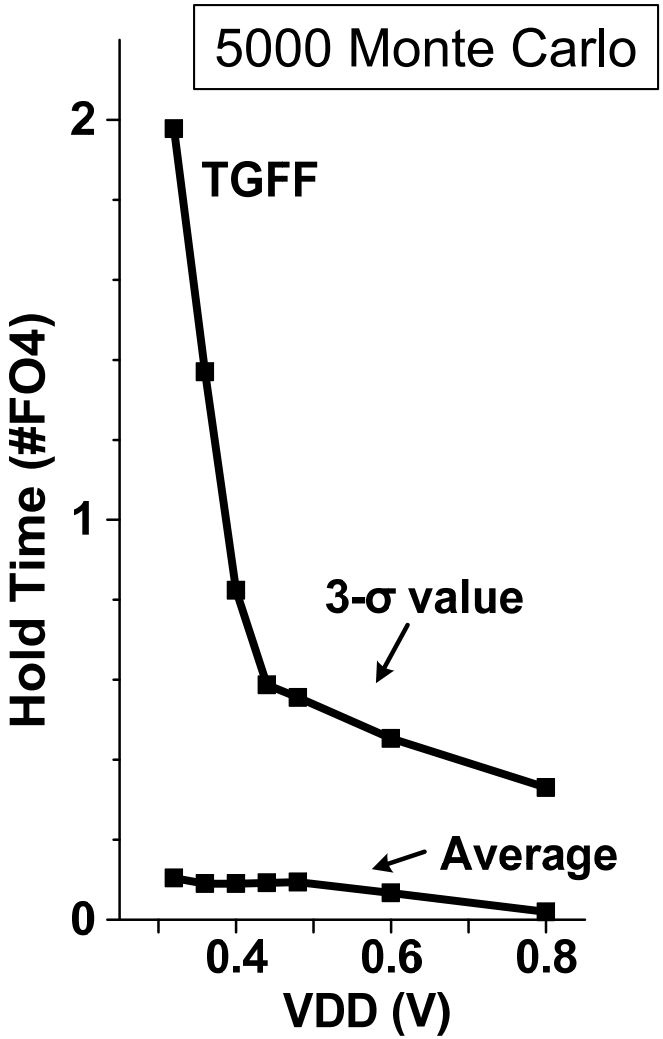



- **Worst-Case $T_{HOLD}$ @ D: 1→0**
- $T_{HOLD}$ dictated by **mismatches** among **I1**, **I3**, **I4**.
- Degraded $T_{HOLD}$ variation at **Low $V_{DD}$**

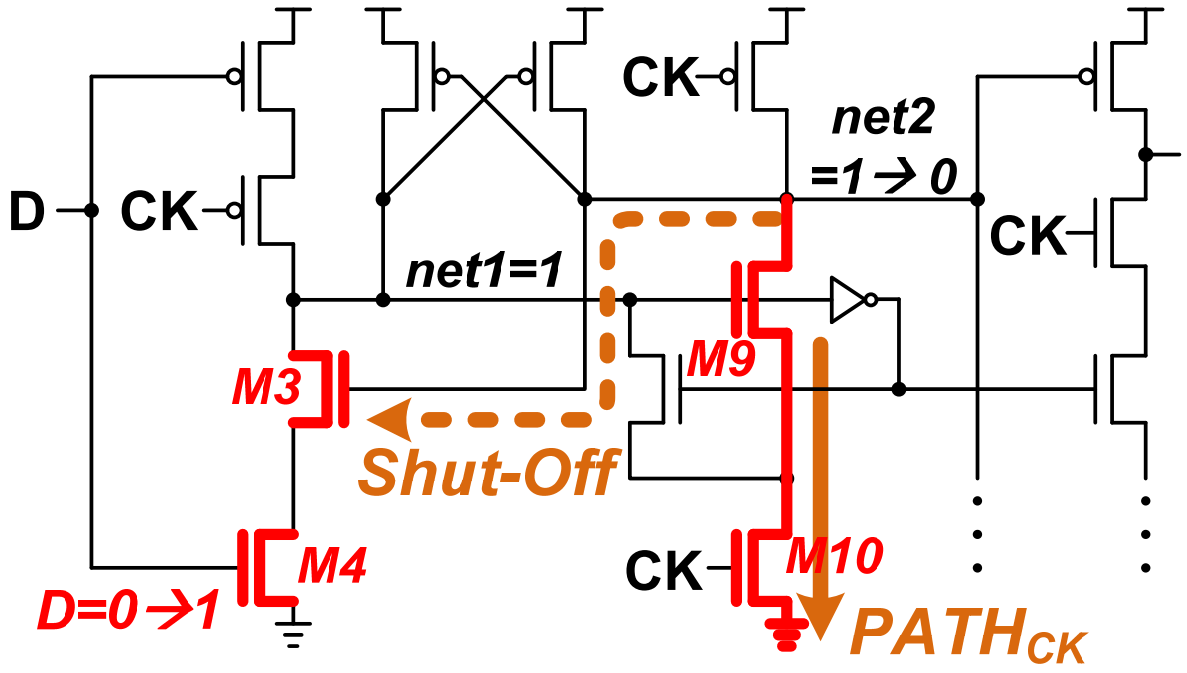
# Worst-Case Hold-Time Path in S²CFF

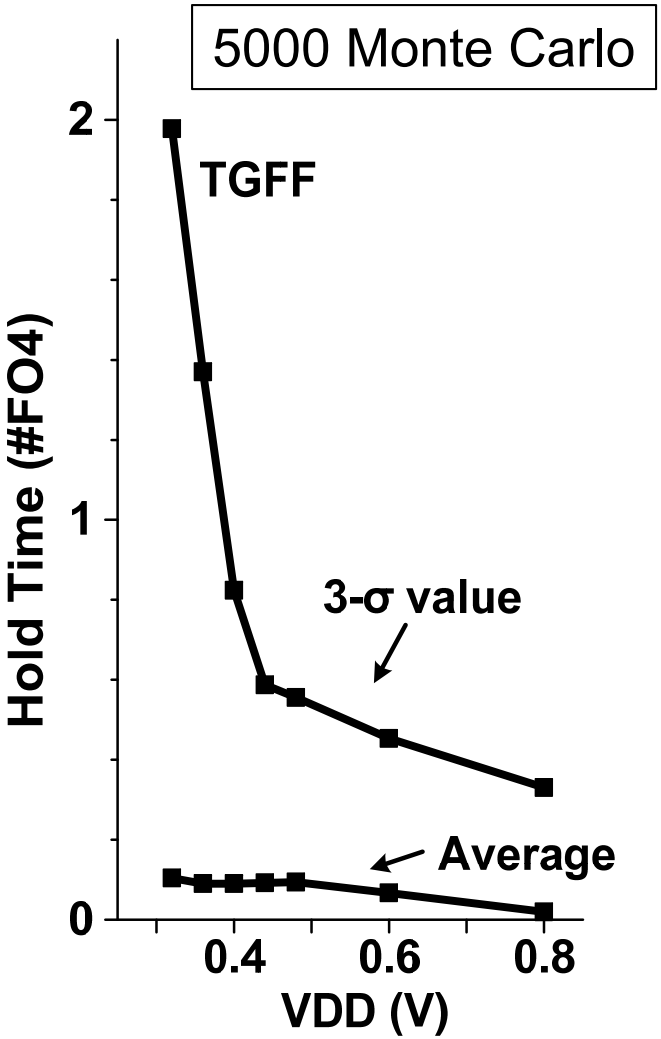



- ***net2*** discharged through **M9/M10**
- ***net2*** shuts-off **M3** ➔ Isolated from **D**

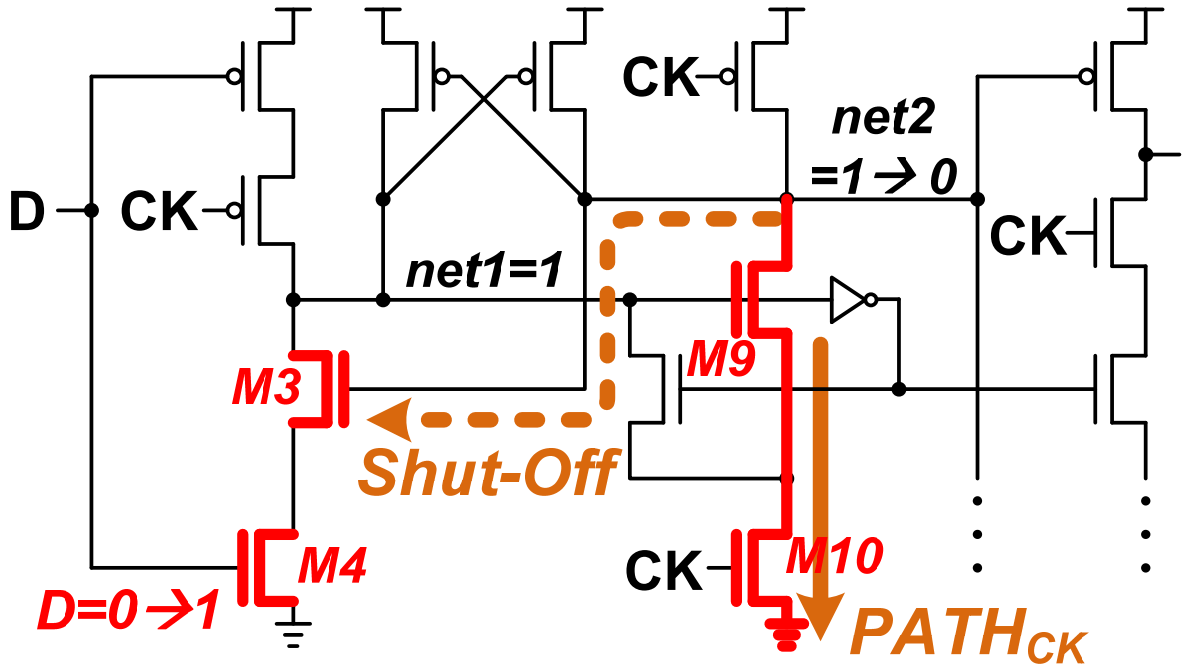
# Worst-Case Hold-Time Path in S²CFF

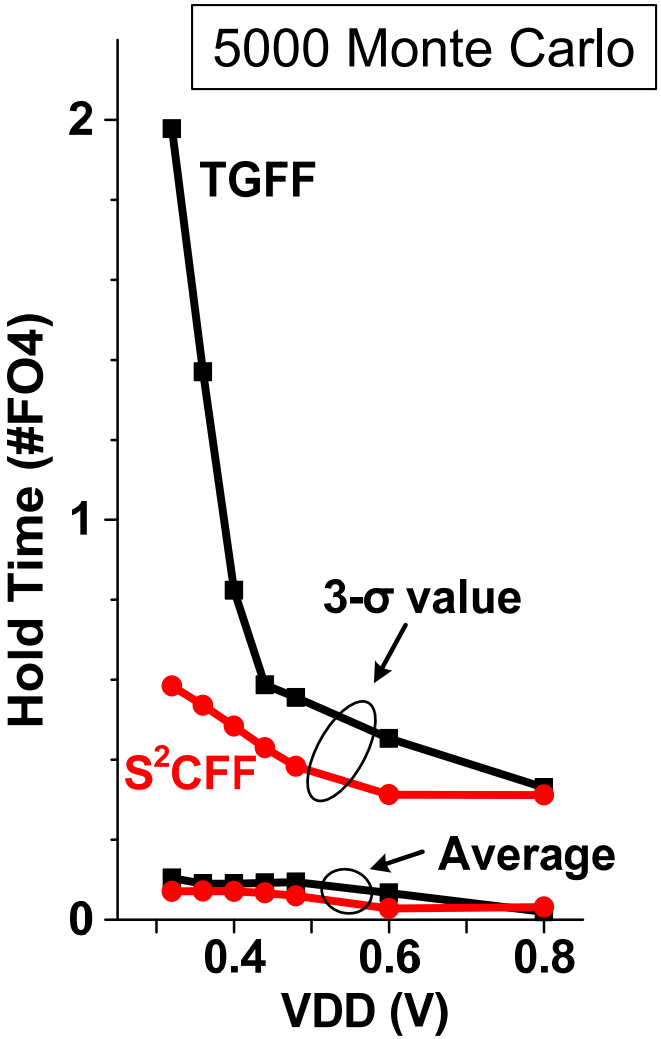



- Discharging speed solely depends on **PATH$_{CK}$**.

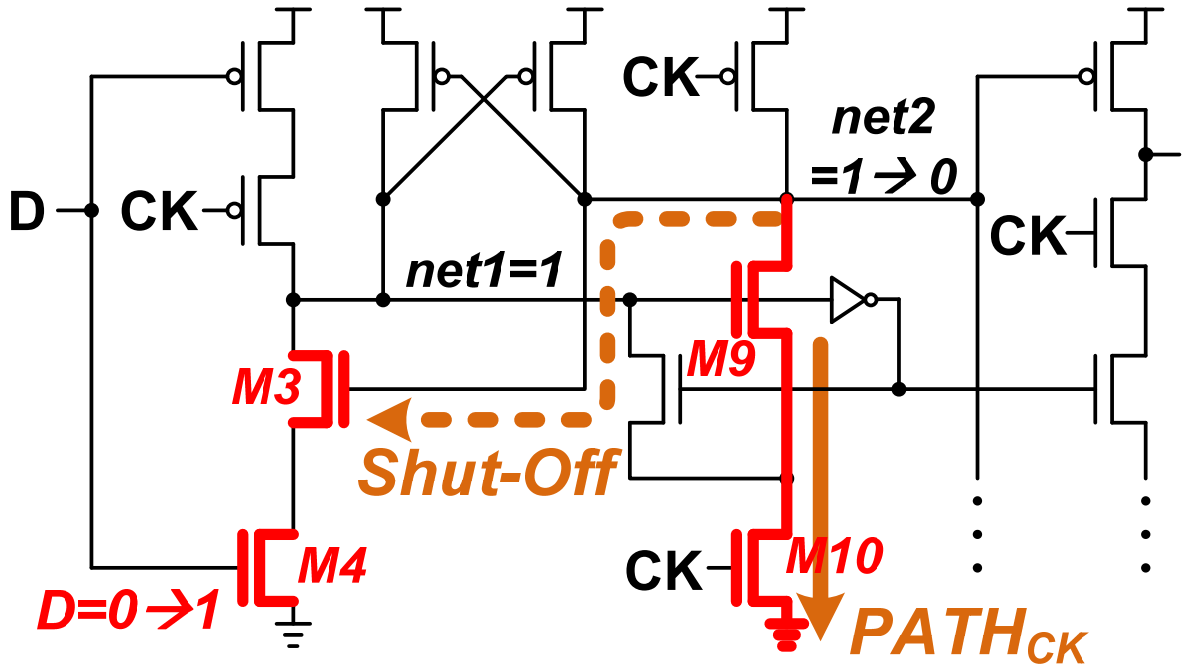
# Worst-Case Hold-Time Path in S²CFF

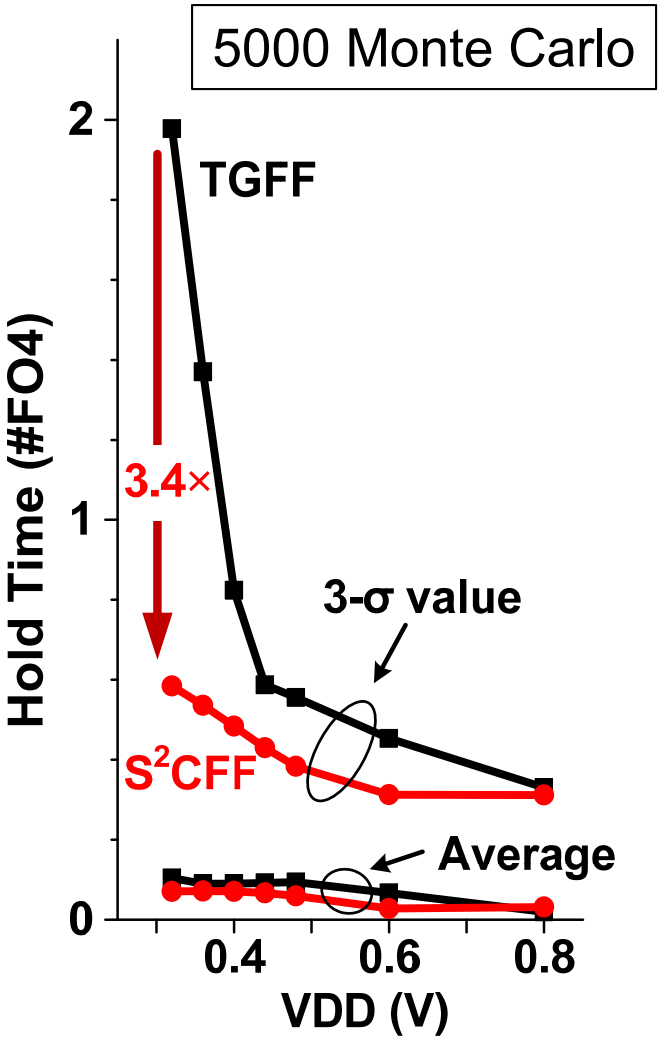



- Discharging speed solely depends on **$PATH_{CK}$**.
- Much better **$T_{HOLD}$** variation

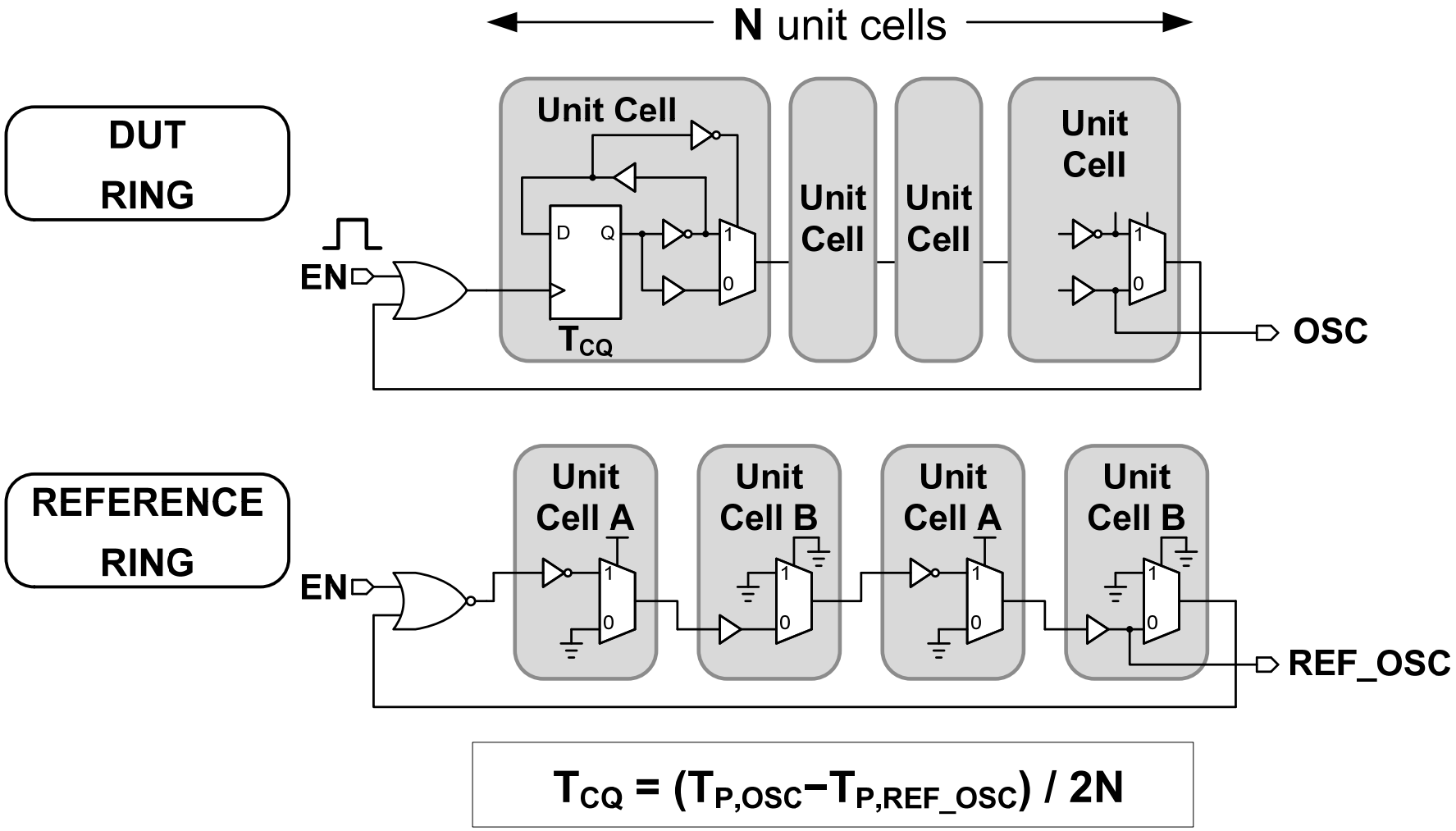
# Worst-Case Hold-Time Path in S²CFF




- Discharging speed solely depends on **$PATH_{CK}$**.
- Much better **$T_{HOLD}$** variation ➔ **3.4x reduction** @ 0.32V

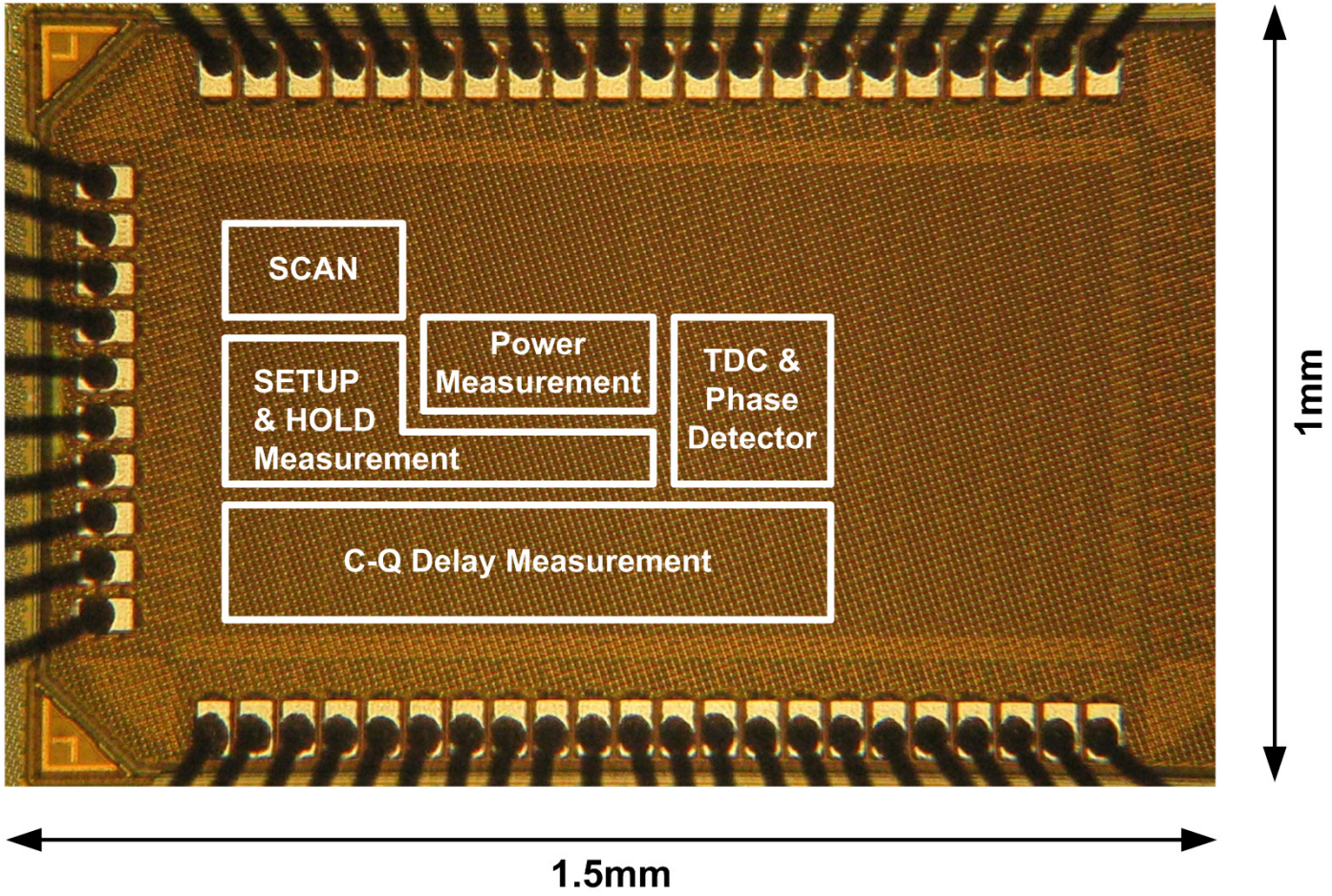
# On-Chip Testing Circuit: C-Q Delay




$$T_{CQ} = (T_{P,OSC} - T_{P,REF\_OSC}) / 2N$$

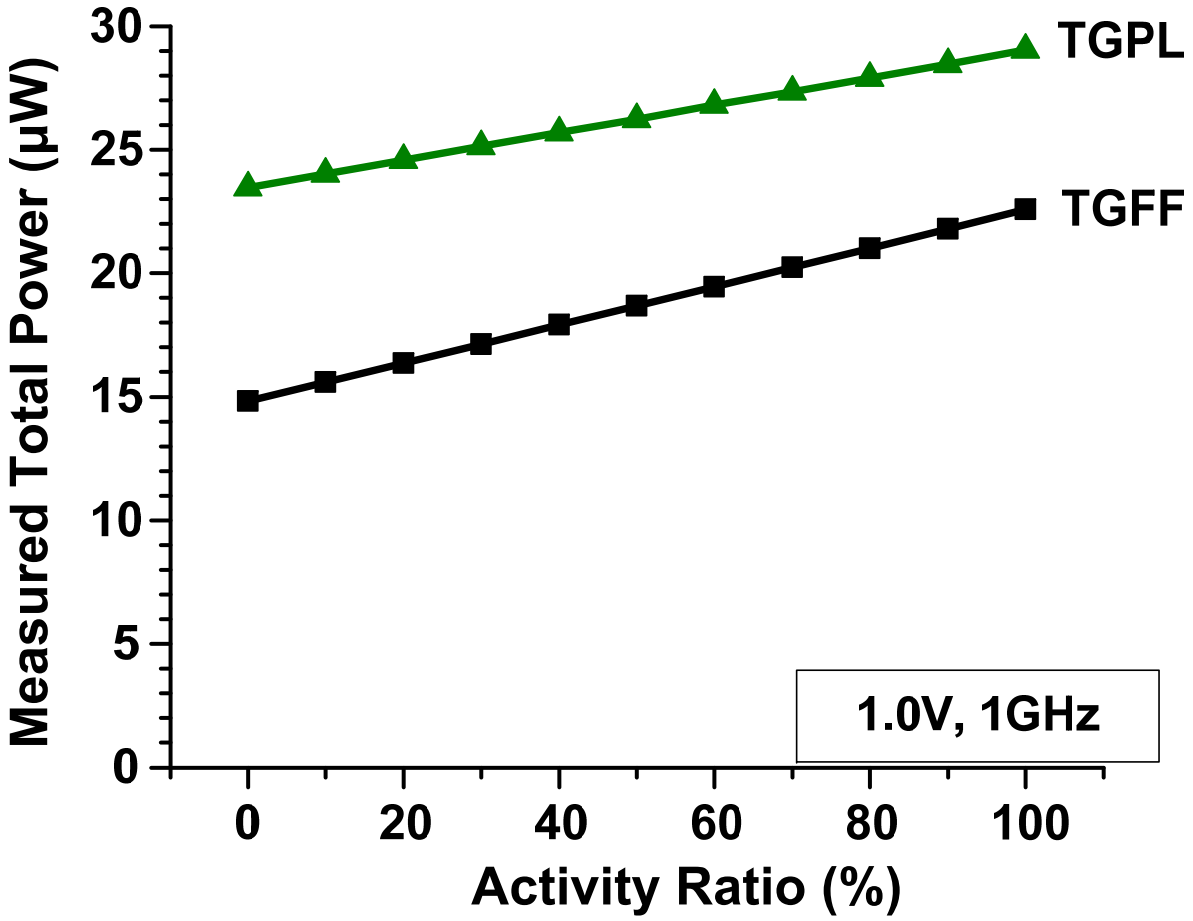
- Large **N** cancels out local mismatches (**N**=100 in the test chip)
- No oscillation if one DUT fails ➔ Provides **Yield Indication**

# Test Chip in 45nm SOI



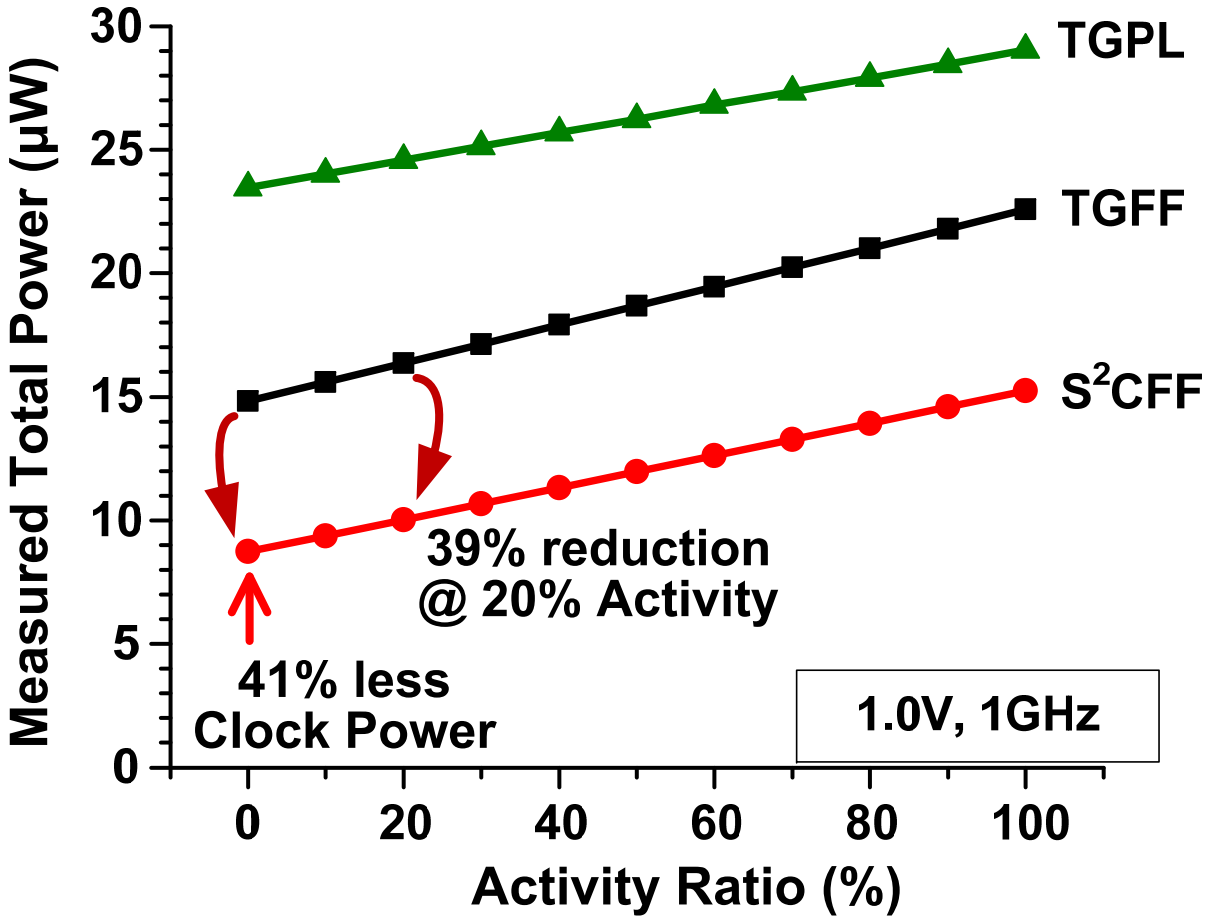

- TGFF, ACFF, TGPL are implemented for comparison
- 40 test chips were measured

# Active Power Measurement @ 1.0V



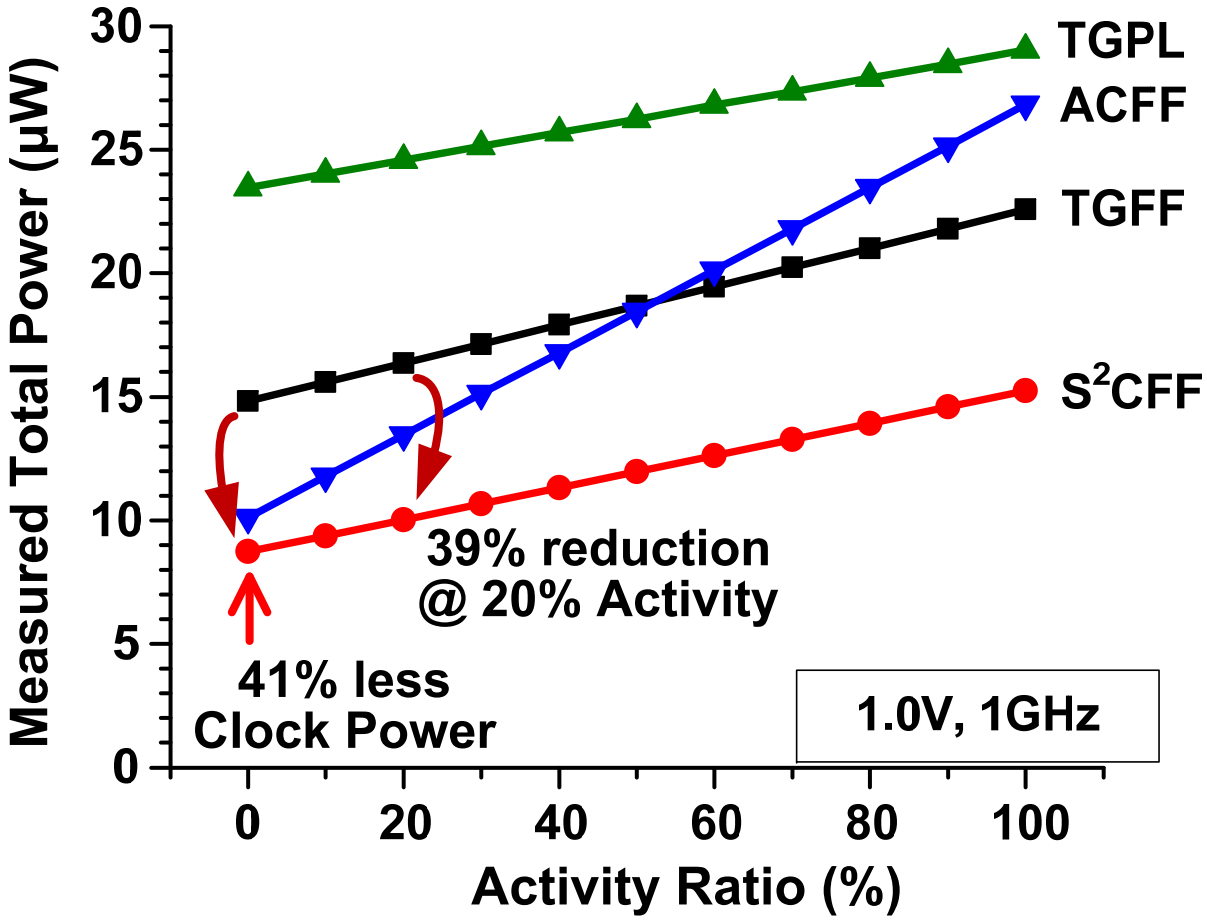

- TGPL shows high power due to Pulse Generation.

# Active Power Measurement @ 1.0V



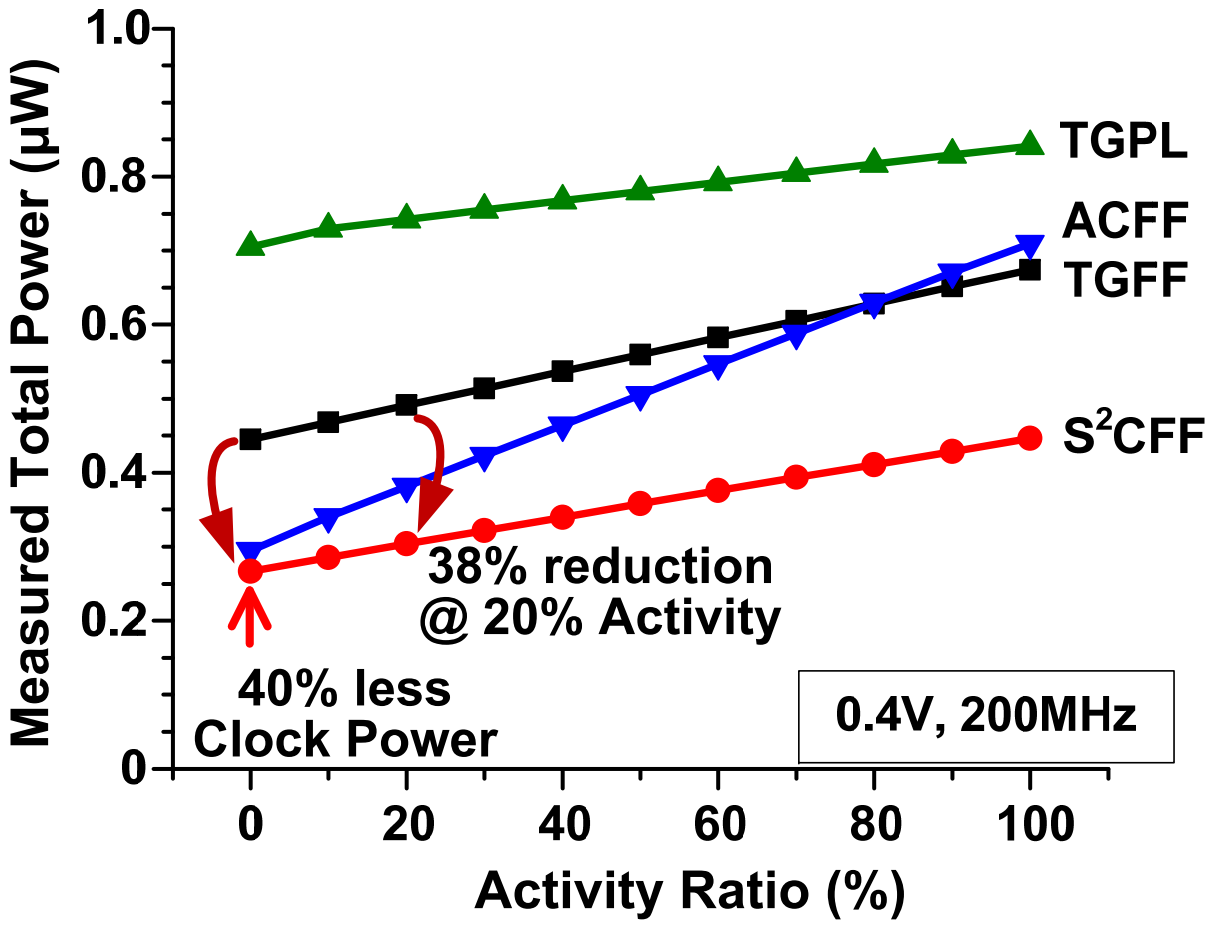

- $S^2$CFF shows 41% and 39% reductions in Clock Power & Total Power, thanks to Single-Phase operation

# Active Power Measurement @ 1.0V



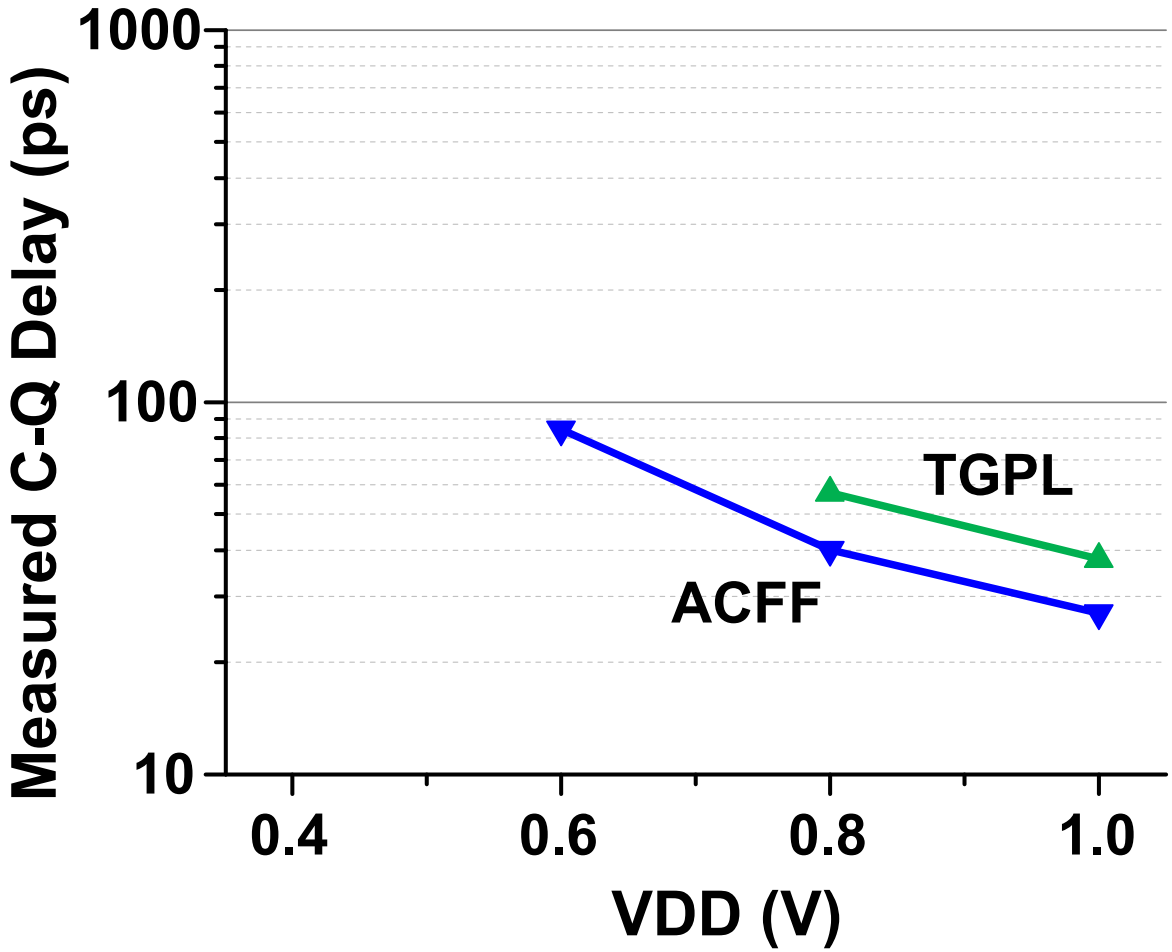

- ACFF total power increases rapidly with activity ratio due to contention

# Active Power Measurement @ 0.4V



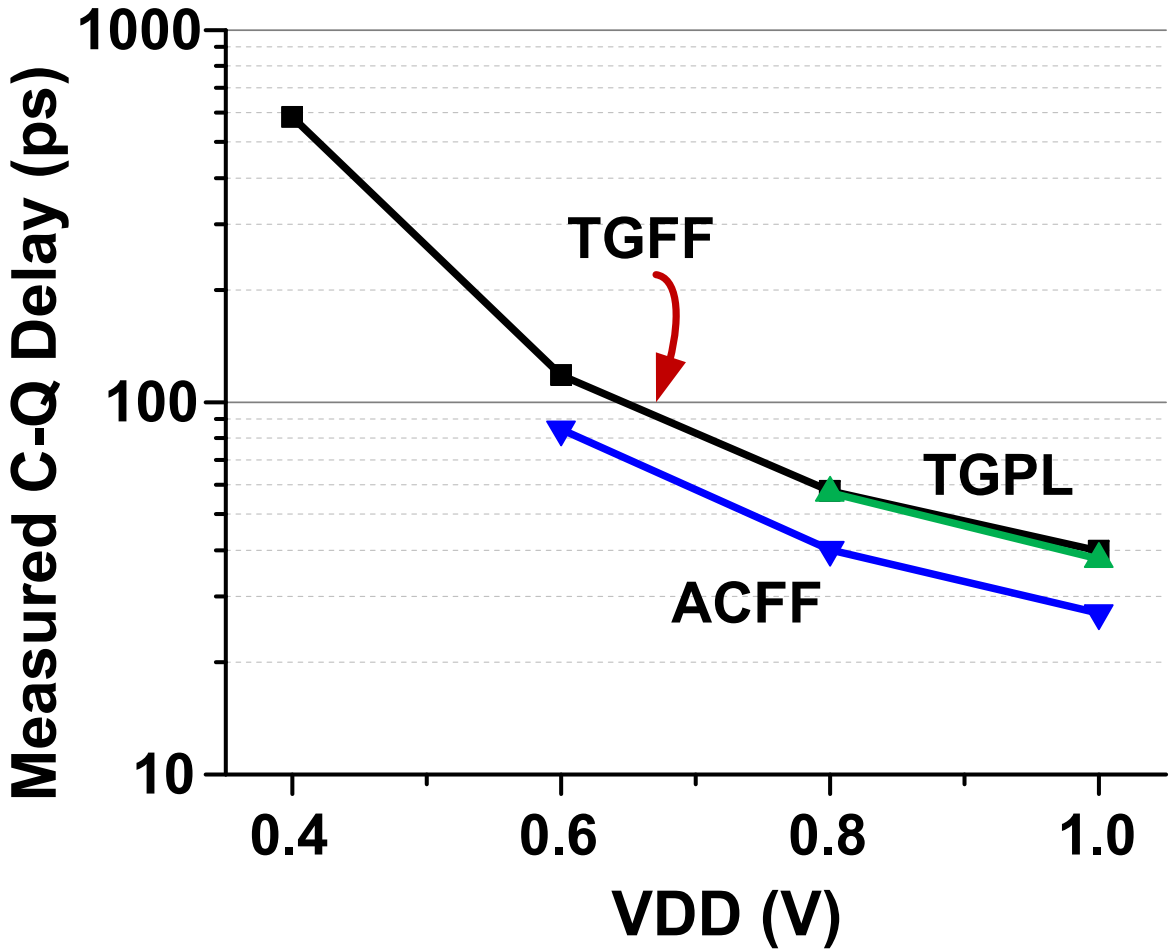

- Similar trend at 0.4V

# C-Q Delay Measurement



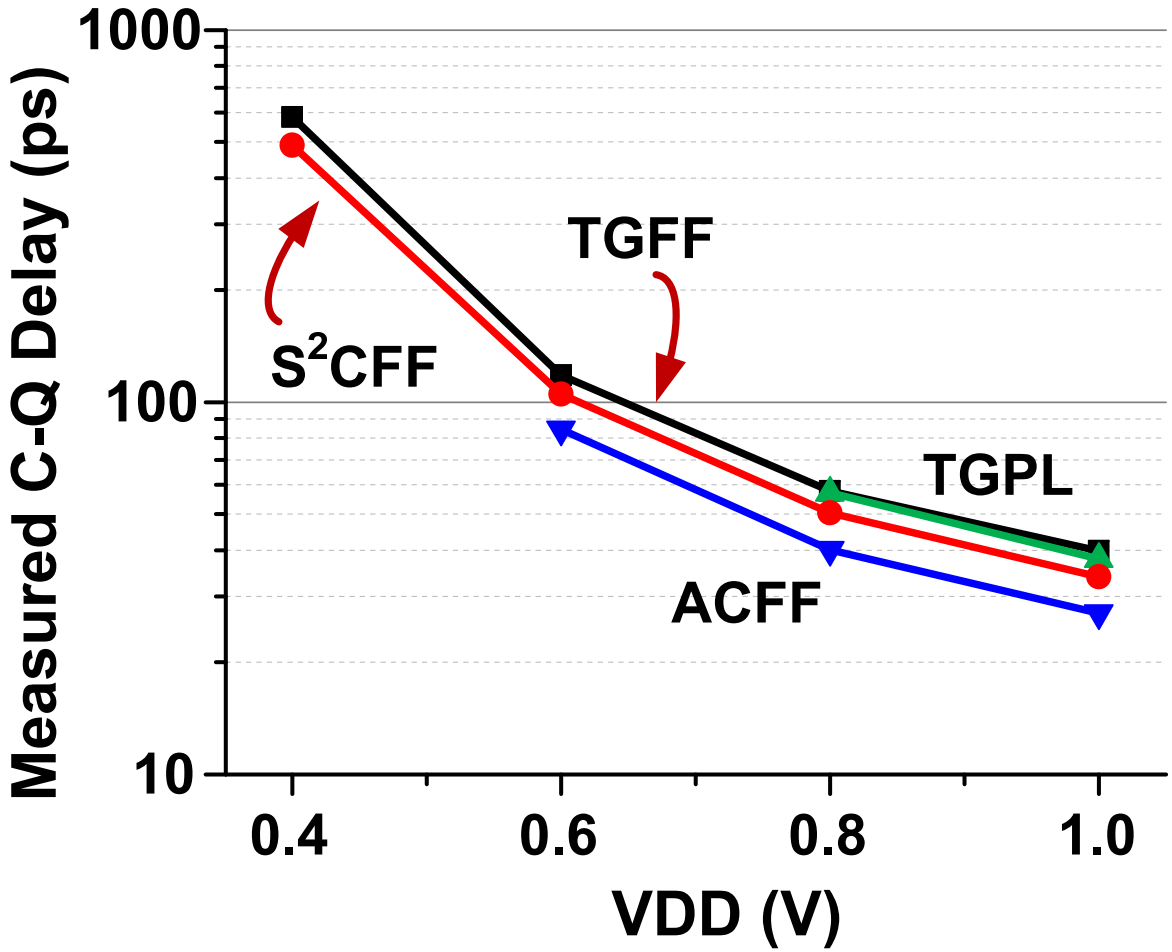

- **TGPL** fails at 0.6V; due to hold time failure
- **ACFF** fails at 0.4V; due to contention

# C-Q Delay Measurement



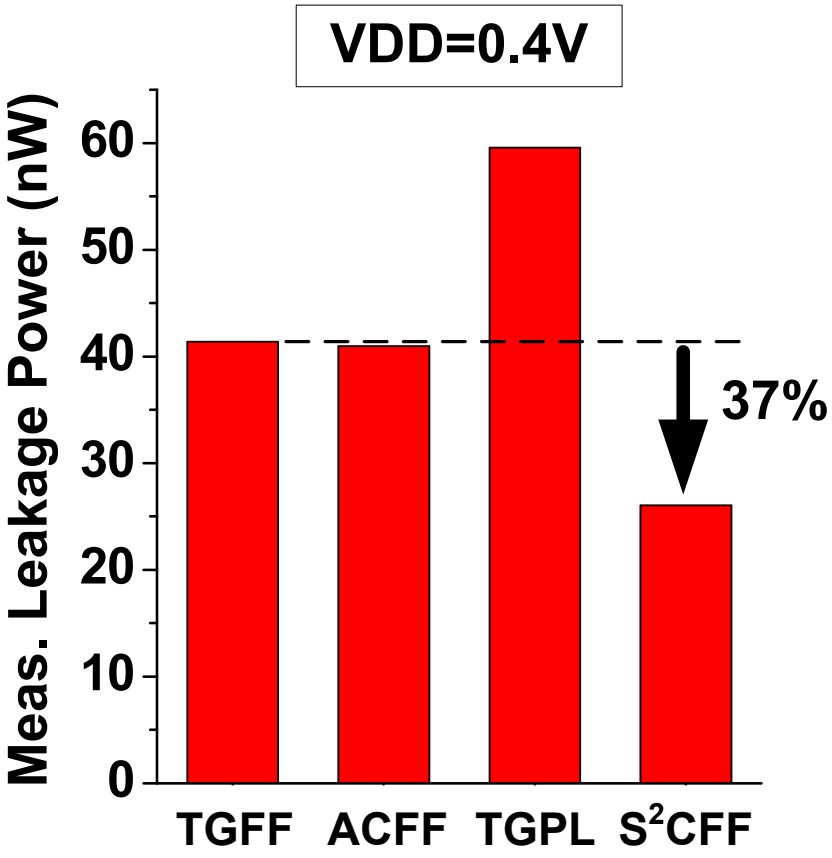

- **TGFF** shows robust operation in the given $V_{DD}$ range thanks to its **Static** and **Contention-Free** operation

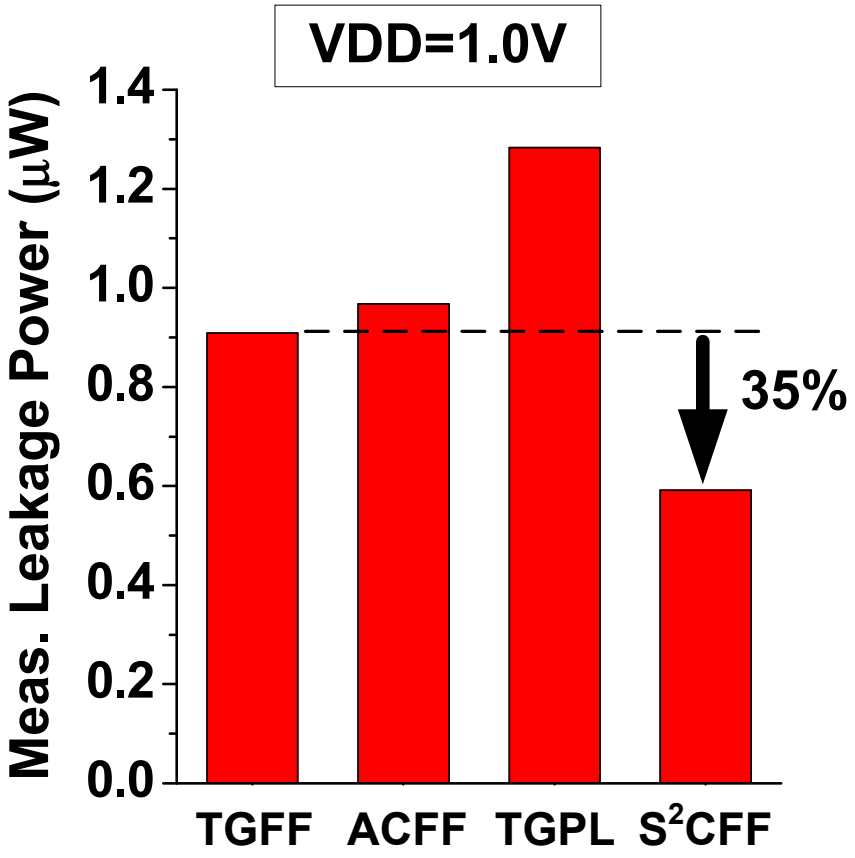
# C-Q Delay Measurement




- **S²CFF** shows robust operation in the given $V_{DD}$ range
- Faster by **14.8%** (1.0V) and **15.9%** (0.4V) vs. TGFF

# Leakage Power Measurement




- **S²CFF** has **35% reduction** in leakage power
  - Fewer leakage paths; most through stacked devices

# Topology Comparison

| | | Type | Contention-Free | Single-Phase Clock | Device Count |
|---|---|---|---|---|---|
| **$S^2CFF$** | This Work | **Static** | **Yes** | **Yes** | **24** |
| **TGFF** | Std Cell Lib. | Static | Yes | No | 24 |
| **ACFF** | ISSCC' 11 | Static | No | Yes | 22 |
| **TGPL** | JSSC' 02 | Pulsed | Yes | No | 28 |
| **$CSP^3L$** | ISSCC' 12 | Pulsed | Yes | No | 42* |
| **DMFF** | ISSCC' 08 | Pulsed | No | No | 24 |
| **CPSA** | ISSCC' 06 | Static | No | Yes | 28 |
| **CCFF** | JSSC' 01 | Pulsed | No | No | 35 |
| **HLFF** | ISSCC' 96 | Pulsed | No | No | 20 |

* 26 excluding pulse generator

# Comparison @ 1.0V

- **ALL Flip-Flops implemented** in the same testchip

| | **S²CFF** This Work | **TGFF** Std Cell Lib | **ACFF** [3] | **TGPL** [4] |
|---|---|---|---|---|
| **Type** | Static | Static | Static | Pulsed |
| **Contention-Free** | Yes | Yes | No | Yes |
| **Single-Phase Clock** | Yes | No | Yes | No |
| **Device Count** | 24 | 24 | 22 | 28 |
| **Layout Size** (Norm.) | 1.07 | 1.00 | 1.13 | 1.40 |
| **C-Q Delay** | 33.9 ps | 39.8 ps | 27.1 ps | 37.9 ps |
| **Setup** (ps) **/ Hold** (ps) | 34.0 / -25.7 | 40.6 / -31.4 | 77.8 / -66.1 | 8.5 / 1.3 |
| **D-Q Delay** (Setup + CQ) | 67.9 ps | 80.4 ps | 104.9 ps | 46.4 ps |
| **Setup + Hold** | 8.3 ps | 9.2 ps | 11.7 ps | 9.8 ps |
| **Total Power** (1GHz, 20% Act.) | 10.0 µW | 16.4 µW | 13.5 µW | 24.6 µW |
| **Leakage Power** | 592 nW | 909 nW | 967 nW | 1283 nW |

# Comparison @ 1.0V

- **ALL Flip-Flops implemented** in the same testchip

| | **S²CFF** This Work | **TGFF** Std Cell Lib | **ACFF** [3] | **TGPL** [4] |
|---|---|---|---|---|
| **Type** | Static | Static | Static | Pulsed |
| **Contention-Free** | Yes | Yes | No | Yes |
| **Single-Phase Clock** | Yes | No | Yes | No |
| **Device Count** | 24 | 24 | 22 | 28 |
| **Layout Size** (Norm.) | **1.07** | **1.00** | **1.13** | **1.40** |
| **C-Q Delay** | 33.9 ps | 39 ps | 27.1 ps | 37.9 ps |
| **Setup** (ps) **/ Hold** (ps) | 34.0 / -2 7 | 40 / -31.4 | 77.8 / -66.1 | 8.5 / 1.3 |
| **D-Q Delay** (Setup + CQ) | | | 104.9 ps | 46.4 ps |
| **Setup + Hold** | | | 11.7 ps | 9.8 ps |
| **Total Power** (1GHz, 20% Act.) | | | 13.5 µW | 24.6 µW |
| **Leakage Power** | 592 nW | 909 nW | 967 nW | 1283 nW |

**7% increase (one poly-pitch)**

# Comparison @ 1.0V

- **ALL Flip-Flops implemented** in the same testchip

| | **S²CFF** This Work | **TGFF** Std Cell Lib | **ACFF** [3] | **TGPL** [4] |
|---|---|---|---|---|
| **Type** | Static | Static | Static | Pulsed |
| **Contention-Free** | Yes | Yes | No | Yes |
| **Single-Phase Clock** | Yes | No | Yes | No |
| **Device Count** | 24 | 28 | 22 | 28 |
| **Layout Size** (Norm.) | 1.07 | 1.00 | 1.13 | 1.40 |
| **C-Q Delay** | 33.9 ps | 39.8 ps | 27.1 ps | 37.9 ps |
| **Setup** (ps) **/ Hold** (ps) | 34.0 / -25.7 | 40.6 / -31.4 | 77.8 / -66.1 | 8.5 / 1.3 |
| **D-Q Delay** (Setup + CQ) | **67.9 ps** | **80.4 ps** | **104.9 ps** | **46.4 ps** |
| **Setup + Hold** | 8.3 ps | 9.2 ps | 11.7 ps | 9.8 ps |
| **Total Power** (1GHz, 20% Act.) | 10.0 µW | 16.4 µW | 13.5 µW | 24.6 µW |
| **Leakage Power** | 592 nW | 909 nW | 967 nW | 1283 nW |

**15.5% faster**

# Comparison @ 1.0V

- **ALL Flip-Flops implemented** in the same testchip

| | **S²CFF** This Work | **TGFF** Std Cell Lib | **ACFF** [3] | **TGPL** [4] |
|---|---|---|---|---|
| **Type** | Static | Static | Static | Pulsed |
| **Contention-Free** | Yes | Yes | No | Yes |
| **Single-Phase Clock** | Yes | No | Yes | No |
| **Device Count** | 24 | 24 | 22 | 28 |
| **Layout Size** (Norm.) | 1.00 | 1.00 | 1.13 | 1.40 |
| **C-Q Delay** | 33.9 ps | 39.8 ps | 27.1 ps | 37.9 ps |
| **Setup** (ps) **/ Hold** (ps) | 34.0 / -33.7 | 40.6 / -31.4 | 77.8 / -66.1 | 8.5 / 1.3 |
| **D-Q Delay** (Setup + CQ) | 67.9 ps | 80.4 ps | 104.9 ps | 46.4 ps |
| **Setup + Hold** | **8.3 ps** | **9.2 ps** | **11.7 ps** | **9.8 ps** |
| **Total Power** (1GHz, 20% Act.) | 10.0 µW | 16.4 µW | 13.5 µW | 24.6 µW |
| **Leakage Power** | 592 nW | 909 nW | 967 nW | 1283 nW |

**10% improvement**

# Comparison @ 1.0V

- **ALL Flip-Flops implemented** in the same testchip

| | **S²CFF** This Work | **TGFF** Std Cell Lib | **ACFF** [3] | **TGPL** [4] |
|---|---|---|---|---|
| **Type** | Static | Static | Static | Pulsed |
| **Contention-Free** | Yes | Yes | No | Yes |
| **Single-Phase Clock** | Yes | No | Yes | No |
| **Device Count** | 24 | 24 | 22 | 28 |
| **Layout Size** (Norm.) | 1.07 | 1.00 | 1.13 | 1.40 |
| **C-Q Delay** | 33.9 ps | 39.8 ps | 27.1 ps | 37.9 ps |
| **Setup** (ps) **/ Hold** (ps) | 34.0 / -25.7 | 40.6 / -31.4 | 77.8 / -66.1 | 8.5 / 1.3 |
| **D-Q Delay** (Setup + CQ) | 67.9 ps | 80.4 ps | 104.9 ps | 46.4 ps |
| **Setup + Hold** | 8.3 ps | 9.2 ps | 11.7 ps | 9.8 ps |
| **Total Power** (1GHz, 20% Act.) | **10.0 µW** | **16.4 µW** | **13.5 µW** | **24.6 µW** |
| **Leakage Power** | **592 nW** | **909 nW** | **967 nW** | **1283 nW** |

# Conclusion: $S^2CFF$

- A New **Flip-Flop** that provides all of the following:
  - **Static** Operation
  - **Contention-Free** Operation
  - **Single-Phase** Clock Operation
  - **Same Device Count** as **TGFF** (**Min. Area Penalty**)

- This results in a **robust** flip-flop that has **~40% lower power** consumption compared to **TGFF**

- **Bonus**: Potential benefit in **$T_{HOLD}$ variation**